Quantum Computing

Contents

1 Introduction

Quantum computing is based on the idea of quantum bit ("qubits" or "Qbits," in Mermin's version). To understand how qubits can compute we'll have to understand wht quantum mechanics says about them. But the term "qubit," like the more familiar term "bit," doesn't denote a physical object; bits can be constructed from many materials. Thus, we'll have to understand how quantum mechanics works in a general way, so that our ideas make sense no matter what physical objects compose qubits. We won't need a lot of details because we won't have to understand how electrons or protons behave. We just need enough to know how qubits are designed to work. Something similar is true for ordinary computation: We don't need to understand semiconductors in order to use a computer, we just need a command of Boolean logic. That's enough to understand circuits and algorithms, no matter how the bits that perform the computations are constructed.

To understand how quantum mechanics works in general, though, it's helpful to look briefly at a physical system that can be used to represent bits. We'll look at photons, and in particular at photon polarization.

2 Polarization, Photons, State Vectors, and Qubits

2.1 E&M

By the end of the 19th century the ideas underlying classical mechanics had been applied to electricity and magnetism. First, electric charge, a property that causes objects to attract or repel other charged objects, was discovered. When charged objects were still, they behaved like gravitating (or antigravitating) things, but when they were moving (as in an electric current) their effects on each other were much more complicated. Eventually Newton's laws of motion for charged objects were supplemented by Maxwell's equations for electric and magnetic fields. These are invisible force fields created at one place by a set of charges and acting at later times on other charges somewhere else. In other words, the fields are produced by charges and their motion, and in turn apply forces to other charges to produce future motion, which produces future fields, etc. In addition, changing fields can affect each other directly even if there are no charges around. If you shake a charged object you produce both changing electric and changing magnetic fields. A change in the magnetic field causes a future change in the electric field and vice versa. What that means is that by shaking charges you can get a self-propagating electromagnetic wave that looks like this:

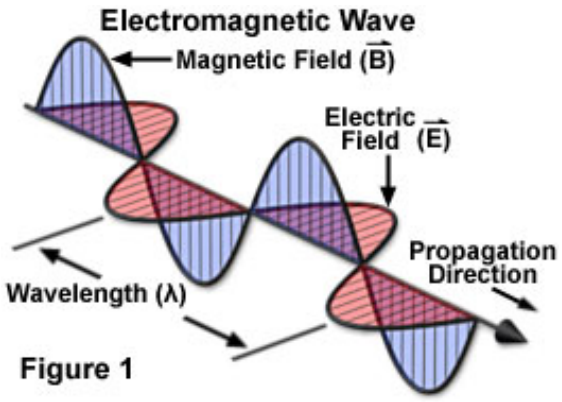
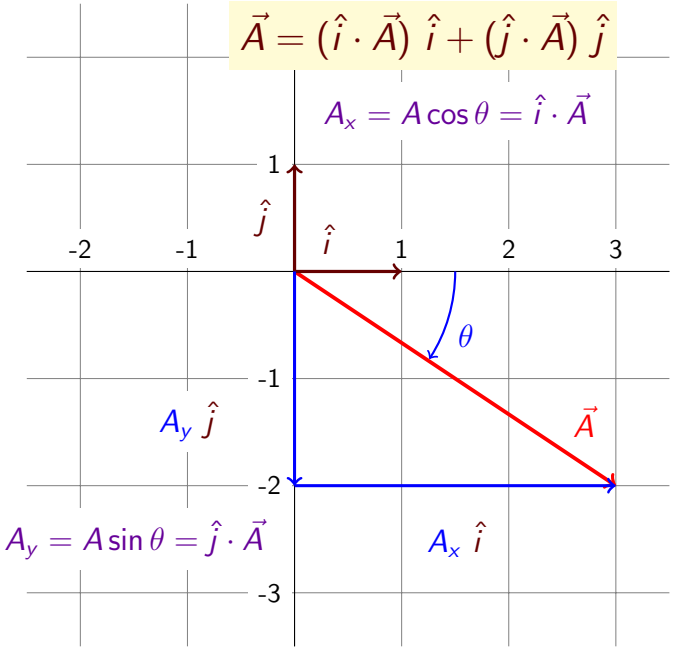


Figure 1

(where the amplitude of the field is given by the height of the red or blue curve) and moves through empty space as in this simulation, or more accurately (for the electric field only) in this other simulation. Light, in this way of looking at it, is a wave like in the picture above with a wavelength between about 390 and 700 nanometers. The waves in these pictures/simulations, by the way, are "linearly polarized," meaning that the electric field vectors always point in the same direction. (Such waves don't have to be; the axis along which the arrows point can change with time). The behavior of polarization when the amplitudes of the fields becomes super small will be our gateway into quantum mechanics.

2.2 Classical Polarization and Photons

2.2.1 Some introductory material on real vectors in the plane



default

Here, the really important facts, which you have to understand, are that

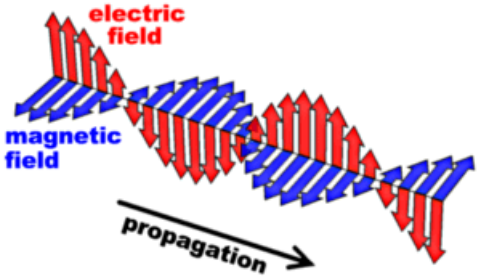
$$\begin{aligned} A_x &= \hat{\imath}\cdot\vec{A} \\ A_y &= \hat{\jmath}\cdot\vec{A} \\ \left(\hat{\imath}\cdot\vec{A}\right)^2 + \left(\hat{\jmath}\cdot\vec{A}\right)^2 &= |\vec{A}|^2 . \end{aligned} \tag{2.2.1}$$

And since there is nothing special about the directions picked out by $\hat{\imath}$ and $\hat{\jmath}$, for any orthogonal unit vectors $\hat{e}_1$ and $\hat{e}_2$ and for any vector $\vec{A}$ with $\vec{A} = A_1\hat{e}_1 + A_2\hat{e}_2$,

$$\begin{aligned} A_1 &= \hat{e}_1\cdot\vec{A} \\ A_2 &= \hat{e}_2\cdot\vec{A} \\ \left(\hat{e}_1\cdot\vec{A}\right)^2 + \left(\hat{e}_2\cdot\vec{A}\right)^2 &= |\vec{A}|^2 . \end{aligned} \tag{2.2.2}$$

2.2.2 Linear Polarization

Maxwell's equations for electric and magnetic fields turn out to imply that both these (vector) fields must always be perpendicular to each other and to the direction of motion of the wave, as in the figure I showed earlier and in the slightly different version here:



Let's suppose the wave is moving in the z direction. At any given point, say the origin, the oscillating wave produces a time-dependent electric field $\mathscr{E}$ given by

$$\vec{\mathscr{E}}(t) = \vec{E}\cos(\omega t + \varphi), \tag{2.2.3}$$

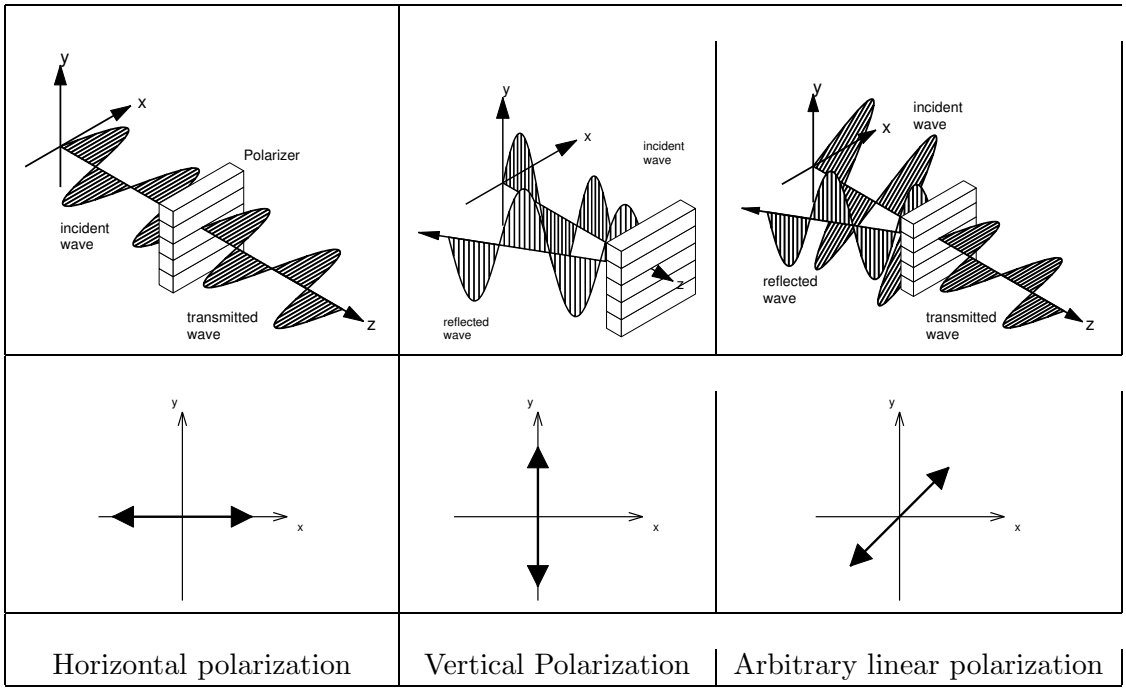
Note the constant vector $\vec{E}$ in front; the full time-dependent field $\vec{\mathscr{E}}$ is always proportional to it. Since $\vec{\mathscr{E}}$ is always perpendicular to the direction of propagation, we have $E_z = 0$ and, we have

$$\vec{E} = E_x\hat{\imath} + E_y\hat{\jmath} \doteq \begin{pmatrix} E_x \\ E_y \end{pmatrix}. \tag{2.2.4}$$

(Note that in the figure above, it looks like $E_x = 0$; the arrows pointing in the x direction denote the magnetic field, not the electric field.) The dotted equals sign here means I can represent the vector by its components in a column as shown. It's not a pure equals sign because the components depend on the set of unit vectors that I choose. In the above, I've chosen unit vectors that point along the x and y axes, i.e. I've represented $\vec{E}$ as the sum of two vectors, one pointing in the x direction and one in the y direction, as in the figure we looked at earlier. I could break up the field into sums in other ways, e.g. as the sum of a field pointing at a positive 45-degree angle and one pointing at a negative

45-degree angle, or as a sum of three vectors, or more. Such representations of the field are called "superpositions."

Now let's look at what happens when this kind of light passes through a polaroid like the one here.



The polaroid is just designed to let through the component of $\vec{\mathscr{E}}$ along a particular direction. I can orient it so that it only lets through the x component of the electric field, as in the figure, or the y component, or some other component. If I orient it so that only the x component gets through, then in passing through the polaroid the constant vector $\vec{E}$ changes as follows:

$$\vec{E} = E_x\hat{\imath} + E_y\hat{\jmath} \quad \xrightarrow{\text{passing through}} \quad E_x\hat{\imath} \;=\; \boxed{(\hat{\imath}\cdot\vec{E})\,\hat{\imath}}\,. \tag{2.2.5}$$

where the boxed expression is the one we will generalize. If I turn the polaroid 90 degrees, we get

$$\vec{E} \quad \xrightarrow{\text{passing through}} \quad E_y\hat{\jmath} \;=\; (\hat{\jmath}\cdot\vec{E})\,\hat{\jmath}\,. \tag{2.2.6}$$

More generally, if I orient the polaroid at some angle θ with respect to the z axis, then, defining $\hat{P} = \cos\theta\,\hat{\imath} + \sin\theta\,\hat{\jmath}$ as a unit vector that points in the direction specified by θ, the vector $\vec{E}$ changes in passing through as follows:

$$\vec{E} \quad \xrightarrow{\text{passing through}} (\hat{P}\cdot\vec{E})\,\hat{P} \quad = \quad \big(E_x\cos\theta + E_y\sin\theta\big)\,\hat{P}\,. \tag{2.2.7}$$

The energy in an electromagnetic wave is proportional to the square of the electric field, so in general, the percentage of of energy transmitted through the polaroid will be

$$F = (\hat{P}\cdot\vec{E})^2/|\vec{E}|^2 \tag{2.2.8}$$

where F here stands for "Fraction of energy." (I've used the fact that the magnitude/length of $(\hat{P}\cdot\vec{E})\hat{P}$ is just $\hat{P}\cdot\vec{E}$.) Note that F is independent of the strength of the field. We can define a unit "polarization vector"

$$\hat{p} \equiv \vec{E}/|\vec{E}| \tag{2.2.9}$$

that points in the direction of the electric field. Then we just have

$$F = \left(\hat{P}\cdot\hat{p}\right)^2 = \text{ square of cos of angle between } \hat{P} \text{ and } \vec{E}\,. \tag{2.2.10}$$

2.2.3 Photons and Quantum Mechanics

OK, now let's consider what happens when individual photons go through a polaroid one at a time. We know that a classical wave is really many many many photons packed close together. These many photons have to give the classical behavior above. So we must have, for linearly polarized light going through a polaroid: $F = (\hat{P}\cdot\hat{p})^2$. This means that a fraction F of the photons must go through the polaroid and $1-F$ must be reflected. But suppose we now make the electric field $\vec{E}$ very small, by making the light very dim. If we want, we can make it so dim that a photon comes along every hour. What does each photon do? They're all identical and doing the exact same thing as they approach the polaroid, yet some go through and some don't. This fact already tells us that the theory of photons must be probabilistic. We can't say what an individual photon will do, we can only specify it had a probability

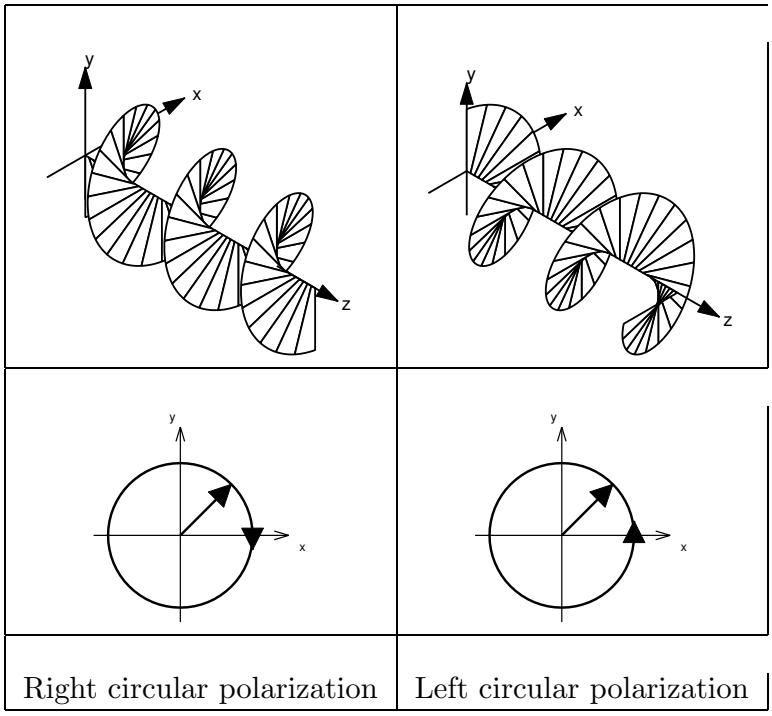
$$\text{Prob.} = F = \left(\hat{P}\cdot\hat{p}\right)^2 \tag{2.2.11}$$

of going through.

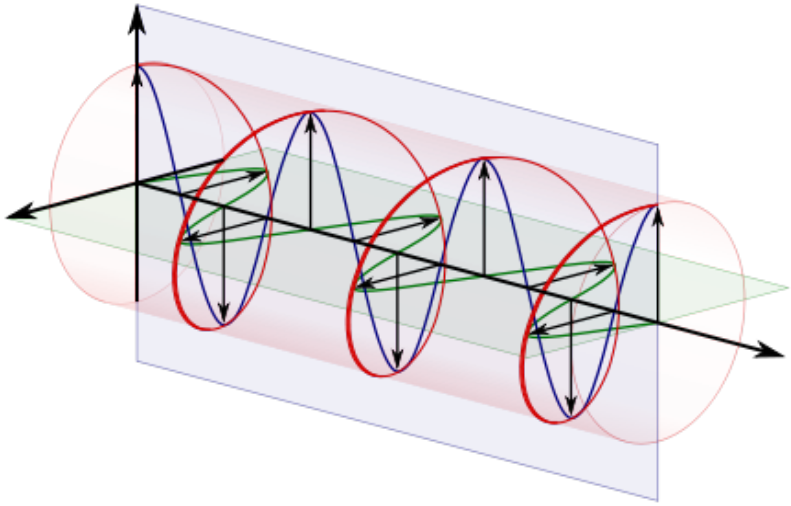
You might ask what the polarization of a single photon even means. After all, a single photon isn't going to cause a measurable oscillation of charges like a ton of them do when a polarized radio wave hits an antenna. We can just take the definition of a horizontally polarized photon to be one that goes through a horizontal polaroid with probability one. A vertically polarized photon is one that is transmitted with probability zero.

2.2.4 A Few Words on Circular Polarization

Linear polarization is not the most general kind. We can have an electric field that propagates along like in this figure,



Right circular polarization | Left circular polarization



with the x and y components out of phase. Then, at any point, we'll have

$$\begin{aligned} E_x &= E_x \cos(\omega t + \varphi) \\ E_y &= E_y \cos(\omega t + \varphi + \alpha)\,, \end{aligned} \tag{2.2.12}$$

for some nonzero α. If $E_x = E_y$ and $\alpha = -\frac{\pi}{2}$ we have "right circular polarization" (the field seems to be rotating clockwise to an observer looking at the wave from behind, and if $E_x = E_y$ and $\alpha = +\frac{\pi}{2}$ we have left circular polarization. If the two E's are unequal or if α is not a multiple of $\pi/2$, we have something called "elliptical polarization." Suffice to say for now that one can conveniently represent circular polarization by unit vectors

$$\hat{p}_L \doteq \frac{1}{\sqrt{2}}(i\,\hat{\imath} + \hat{\jmath}) \doteq \begin{pmatrix} i \\ 1 \end{pmatrix}, \qquad \hat{p}_R \doteq \frac{1}{\sqrt{2}}(-i\,\hat{\imath} + \hat{\jmath}) \doteq \begin{pmatrix} -i \\ 1 \end{pmatrix}, \tag{2.2.13}$$

where the i in front of $\hat{\imath}$ is defined to be $i = \sqrt{-1}$. To make these things unit vectors, we have to change the definition of the dot product, which we use to define length, so that it works with complex vector components. Once we do that, we can still express the probability of passing through a polaroid or prism as in (2.2.11). We'll see how that works shortly.

2.2.5 Quantum Mechanics

We've now have an idea of how QM works, at least for photon polarization. Photons don't in general have a definite polarization in a particular direction or circular orientation. They are in some definite polarization state, i.e. there is a (perhaps complex) vector that describes their polarization, but that vector is usually not the same vector that describes the orientation of the polarizer, prism, beam-splitter, or other device that is able to measure polarization. For any polarization measurement there are two possible outcomes (transmission or reflection/absorption) and the two polarization vectors corresponding to the transmitted or reflected photons are always perpendicular (or, for circular/elliptial polarization, the generalization of perpendicular that we'll come to). So we can have a beam splitter that spits $\uparrow$ and $\rightarrow$ polarization, or $\nearrow$ and $\nwarrow$ polarization, but not $\uparrow$ and $\nearrow$.

Because photons can have complex polarization vectors that can't be drawn as arrows, and also because qubits don't have to be associated with photon at all, we will abandon the notation $\vec{v}$ for vectors and replace it with something like $|v\rangle$, called a "ket" by Dirac, for reasons we'll see later. We'll denote the quantum polarization state of a vertically photon $|\uparrow\rangle$, a horizontally polarized photon $|\rightarrow\rangle$, a 45°-polarized photon $|\nearrow\rangle$, a $-45°$ polarized photon $|\nwarrow\rangle$, a right-circularly polarized photon by $|\circlearrowright\rangle$ and a left circularly-polarized photon by $|\circlearrowleft\rangle$.

Photons can represent quantum bits. We might take the polarization state $|\uparrow\rangle$ to represent the qubit 0, so that we can write

$$|0\rangle \equiv |\uparrow\rangle\,, \qquad |1\rangle \equiv |\rightarrow\rangle\,. \tag{2.2.14}$$

Then what is the meaning of the state $|\nearrow\rangle$? If the photon is in this state, it represents neither 0 or 1, but rather a "superposition" of the two. "Superposition" might seem like a vague word, but it's mathematically precise in that

$$|\nearrow\rangle = \frac{1}{\sqrt{2}}\left(|\uparrow\rangle + |\rightarrow\rangle\right). \tag{2.2.15}$$

Similarly, as you can check for yourself, a vertical polarization state can be represented as superposition of 45° and $-45°$ polarization states:

$$|\uparrow\rangle = \frac{1}{\sqrt{2}}\left(|\nearrow\rangle + |\nwarrow\rangle\right), \tag{2.2.16}$$

or circular-polarization states:

$$|\uparrow\rangle = \frac{1}{\sqrt{2}}\left(|\circlearrowright\rangle + |\circlearrowleft\rangle\right). \tag{2.2.17}$$

Thus, superposition is not an absolute concept. A state can be a superposition of states in some basis while corresponding to a single basis state in another basis.

If a polarization bit is in the state (2.2.15), then when you measure to see whether the bit is a 0 or a 1 you find you get either, with a 50% probability for each. But that doesn't mean that it was a 0 or 1 before the measurement and you just didn't know which. It should be clear that a photon with ↗ polarization is is neither horizontally nor vertically polarized!

Before getting further into qubits, let's go over some of the math we need to describe them more systematically.

2.3 Quantum Mechanical Formalism: State Vectors

The theory of QM is set up so that all the information we can get — the probability for anything to happen when we do something resembling a measurement, including the examination of a qubit to determine its value — is contained in a "state vector." The set of all possible state vectors is generalization of our usual notion of the set of arrows in space. So let's review/learn some vector math to get started. The results will allow us to discuss what happens when you measure the value of a number contained in a quantum register, and how to run those numbers through the quantum equivalent of gates.

2.3.1 Complex numbers

It's important to know a few basic facts about complex numbers, and some basic notation.

1. $i \equiv \sqrt{-1}$.
2. A general complex number can be written as $z = a + ib$, a, b, real. The numbers a and b are the Cartesian coordinates of z in the complex plane.
3. Polar representation $z = re^{i\theta}$, r, θ real. The numbers r and θ are the polar coordinates of z in the complex plane. This fact, which I might ask you to prove, is equivalent to Euler's identity: $e^{i\theta} = \cos\theta + i\sin\theta$.
4. $z^* = \bar{z} \equiv a - ib = re^{-i\theta}$.
5. $|z| \equiv \sqrt{z^* z} = \sqrt{a^2 + b^2} = r$.

2.3.2 What vectors are: definition of vector space

In the following I'll talk about abstract vectors but use the usual arrow vectors in two and three real dimensions as examples.

A vector space, roughly speaking, is defined to be a set of objects (vectors), which we will denote by the symbol $|\rangle$ (a "ket"). A letter or other symbol inside a ket (e.g. $|v\rangle$) denotes a particular vector, in essentially the same way a letter under an arrow does.

Why such a weird symbol for a vector? Here's Mermin's explanation: As we said, , it doesn't really make sense to use arrows, because in quantum mechanics we will usually

be working in more than two or three dimensions, and with vectors that can be "complex." But if you have a basis (we'll get to that) in an N-dimensional space , why not write the basis vectors, as mathematicians do, as something like $\varphi_1, \varphi_2, \ldots, \varphi_N$?. One reason is that for us these vectors will represent a set of quantum bits, which can correspond to numbers. So the subscripts on the φ's are the important things and shouldn't just be subscripts. Thus, we write $|1\rangle, |2\rangle, \ldots |N\rangle$, or in binary, e.g., $|000\rangle, |001\rangle, |010\rangle, \ldots, |111\rangle$. There are more reasons for using this notation; we'll come to them soon.

OK, what are these ket things? Vectors (or kets, or quantum states; they're all the same thing) can be added and multiplied by numbers (complex numbers in our case). Addition is commutative,

$$|a\rangle + |b\rangle = |b\rangle + |a\rangle \,, \tag{2.3.1}$$

and associative

$$(|a\rangle + |b\rangle) + |d\rangle = |a\rangle + (|b\rangle + |d\rangle) \,, \tag{2.3.2}$$

and multiplication is distributive:

$$c\,(|a\rangle + |b\rangle) = c\,|a\rangle + c\,|b\rangle \,. \tag{2.3.3}$$

Both the addition of vectors and the multiplication of vectors by a numbers produce another member of the vector space. There is a zero vector, which we'll just call 0, the same symbol we give to the number 0, defined so that

$$|a\rangle + 0 = |a\rangle \tag{2.3.4}$$

for all $|a\rangle$. Every vector $|a\rangle$ has an inverse -$|a\rangle$, such that

$$|a\rangle + -\,|a\rangle = 0\,. \tag{2.3.5}$$

The existence of an inverse lets us define subtraction of vectors

$$|a\rangle - |b\rangle \equiv |a\rangle + -\,|b\rangle \,. \tag{2.3.6}$$

The set of arrows extending in the plane from the origin — what we think of as vectors — make a good example that satisfies these rules. We can define addition by the "tip-to-tail" rule and multiplication by a real number by scaling the length of the arrow without changing its direction. (We can't, however, multiply by an arrow by an imaginary number and get another arrow). 0 is an arrow with no length, and the inverse of an arrow is just a backwards version of the same arrow. Clearly addition and multiplication by numbers obey the usual commutativity, associativity, and distributivity requirements, and just as clearly they produce other arrows in the plane. The set of all of the arrows in the plane is the vector space.

2.3.3 Linear independence, dimension, and bases

A linear combination of N vectors is another vector of the form

$$|v\rangle = c_1\,|v_1\rangle + c_2\,|v_2\rangle + \cdots + c_N\,|v_N\rangle \,, \tag{2.3.7}$$

for some set of c_i and $|v_i\rangle$. A linearly-independent set of vectors is a set in which none of the vectors can be expressed as a linear combination of the others.

In any finite-dimensional vector space, there is a maximum number of linearly independent vectors you can choose. In the Euclidean plane, for example, a set of linearly independent vectors has at most two elements. In any set of three (or more) vectors, I'll be able to express one (it doesn't matter which) as a linear combination of the other two. Do you see why? The maximum number of linearly independent vectors I can find is defined to be the dimension of the space. So the plane has two dimensions! How many dimensions does the three-dimensional space have???

Note that I have made no mention of components (or coordinates) in defining the dimension of the space. But we'll take up components now. A linearly independent set with d members, where d is the dimension of the space, is one that allows me to write any vector in the space as a linear combination of members of the set. We say that such a set forms a "basis" (or "complete basis") for the vector space. The c's in the combination for a given vector $|v\rangle$ (they are unique) are called its "expansion coefficients" or "components" or "coordinates" in the basis. Note that these coefficients/components depend on the basis. Consider the 2d vector

$$\vec{v} = 2\hat{i} + \hat{j}\,. \tag{2.3.8}$$

Its components in the usual $\hat{i}, \hat{j}$ basis are 2 and 1, so that I can write

$$\vec{v} \doteq \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \tag{2.3.9}$$

where the dotted equals sign, instead of the usual equals sign, is to emphasize that the vector is represented by its components, which can change if the basis is differentj. Consider, for example, theh basis: consisting of $\hat{i}$ and $\hat{i}+\hat{j}$. In this new basis, the components of $\vec{v}$ are 1 and 1, so that

$$\vec{v} \doteq \begin{pmatrix} 1 \\ 1 \end{pmatrix}. \tag{2.3.10}$$

Note that the second basis vector isn't a unit vector and isn't orthogonal to the first one. That doesn't matter; we can define components without worrying about length and orthogonality. Using those concepts makes things easier, though, and we'll take them up now.

2.3.4 Inner product (or dot product)

We'll just define an inner product to be a product of two vectors that gives a number. There are many ways of dong this, but any inner product has to obey special conditions, generalizations of linearity, which we'll see is enough to get the usual expression for them in an orthonormal basis. The conditions are:

Conditions: Denote the inner product of $|v_1\rangle$ and $|v_2\rangle$ by $\langle v_1|v_2\rangle$, where I'm using subscripts in kets, despite what I said about them earlier. Then

$$\begin{aligned} &|A\rangle = c_1\,|v_1\rangle + c_2\,|v_2\rangle \\ &\quad\Longrightarrow \langle B|A\rangle = c_1\,\langle B|v_1\rangle + c_2\,\langle B|v_2\rangle \quad \text{and} \quad \langle A|B\rangle \;\; = \langle B|A\rangle^* \end{aligned} \tag{2.3.11}$$

Is the dot product of two arrows in the plane an inner product according to this definition? Well, it does take two vectors into a number. It turns out that the dot product is a fine inner product that obeys the relation in Eq. (2.3.11). I used to give the proof as a homework problem.

A basis in which every member is orthogonal to every other one, so that $\langle e_i|e_j\rangle = 0$ if $i \neq j$ and for which every member has length 1, so that $\langle e_i|e_i\rangle = 1$, $\forall i$ is called "orthonormal." In such a basis it's very easy to find the components of an arbitrary vector. Since we can write,

$$|v\rangle = c_1|e_1\rangle + c_2|e_2\rangle + \cdots + c_N|e_N\rangle\,, \tag{2.3.12}$$

and since we have $\langle e_i|e_j\rangle = 1$ if $i = j$ and 0 otherwise, we get, e.g.,

$$\langle e_3|v\rangle = 0 + 0 + c_3 + 0\ldots = c_3\,, \tag{2.3.13}$$

and the same for any other component; that is

$$\boxed{c_i = \langle e_i|v\rangle\,.} \tag{2.3.14}$$

As we saw earlier, we're used to applying this result for Euclidean vectors in 2d, where the components along the x and y basis vectors, that is the coefficients A_x and A_y of the vector $\vec{A}$ (draw picture):

$$\vec{A} = A_x\hat{\imath} + A_y\hat{\jmath}\,, \tag{2.3.15}$$

are

$$\begin{aligned} A_x &= A\cos\theta = 1\times A\times\cos\theta = \hat{\imath}\cdot\vec{A} \\ A_y &= A\sin\theta = 1\times A\times\sin\theta = \hat{\jmath}\cdot\vec{A} \quad \checkmark\,, \end{aligned} \tag{2.3.16}$$

where θ is the angle between $\vec{A}$ and the x-axis.

We also have a nice formula for the inner product in terms of components if $|a\rangle = a_1|e_1\rangle + a_2|e_2\rangle + \cdots a_N|e_N\rangle$ and $|b\rangle = b_1|e_1\rangle + b_2|e_2\rangle + \cdots b_N|e_N\rangle$ then

$$\langle a|b\rangle = \left(a_1^*\langle e_1| + a_2^*\langle e_2| + \cdots a_N^*\langle e_N|\right)\left(b_1|e_1\rangle + b_2|e_2\rangle + \cdots b_N|e_N\rangle\right)$$

(you are to show the above in the homework)

$$\begin{aligned} &= a_1^* b_1\langle e_1|e_1\rangle + a_1^* b_2\cancel{\langle e_1|e_2\rangle} + a_1^* b_3\cancel{\langle e_1|e_3\rangle}\cdots \\ &+ a_2^* b_1\cancel{\langle e_2|e_1\rangle} + a_2^* b_2\langle e_2|e_2\rangle + a_2^* b_3\cancel{\langle e_2|e_3\rangle} + \cdots \\ &+ \quad \vdots \\ &= a_1^* b_1 + a_2^* b_2 + \cdots + a_N^* b_N\,. \end{aligned} \tag{2.3.17}$$

This works as long as the basis vectors are orthonormal. It is clearly just a generalization of the component representation of the dot product for Euclidean vectors, right?

Let's spend a little more time on this component representation (always remembering that we're working in a particular orthonormal basis and that the components change when we change bases). As we've seen, we can arrange the components in a column vector:

$$|a\rangle \doteq \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{pmatrix}. \tag{2.3.18}$$

Then we can write the dot product in Eq. (2.3.17) as

$$\langle a|b\rangle = \begin{pmatrix} a_1^* & a_2^* & \cdots a_N^* \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{pmatrix}. \tag{2.3.19}$$

We'll be working almost exclusively with orthonormal bases in this course.

2.4 Quantum Measurement

Quantum mechanics as a physical theory has two parts. The first, in the usual formulation, is the "Schrödinger equation", a relation analogous to Newton's second law, that tells you how quantum states change in time when you're not bothering them. In our context, that will be the part of the theory that specifies how quantum gates and circuits affect collections of qubits. The second part is a completely different law that tells you what happens to a quantum state when you measure some property of the system it represents. Much of the weirdness of quantum mechanics is related to the big differences between these two kinds of laws.

Since so far we've discussed the polarization of single-photons, each of which can represent a single qubit, we'll first look at what happens when you measure a property of a sate in a two-dimensional vector space. Here is how it works. Any basis has two orthogonal unit vectors, which we can call $|u\rangle$ and $|u^\perp\rangle$. Any vector in the space can be written as

$$|v\rangle = \alpha\,|u\rangle + \beta\,|u^\perp\rangle\,. \tag{2.4.1}$$

Suppose you want to measure whether the qubit is in state $|u\rangle$. First, you must make sure that $|v\rangle$ is normalized, i.e. that

$$\langle v|v\rangle = 1\,. \tag{2.4.2}$$

If it's not, you must scale it so that it is, which means redefining α and β. Then:

a) You will find the qubit to be in the state $|u\rangle$ with probability $|\langle u|v\rangle|^2 = |\alpha|^2$, meaning, roughly, that if you have a lot of qubits all in the state $|v\rangle$, that a fraction $|\alpha|^2$ of them will be found to be in $|u\rangle$ when you check. The rest will be found to be in $|u^\perp\rangle$, so that the probability for ending in that state is $|\langle u_\perp|v\rangle|^2 = |\beta|^2$. There are no other possibilities. This statement is called the "Born rule." We'll have to modify it a little for multi-qubit systems.

b) Because the qubit is in the state $|v\rangle$ before the measurement, sometime during the measurement it "collapses" from $|v\rangle$ into $|u\rangle$ (if you've measured it to be in that state) or $|u^\perp\rangle$ (if you've measured it to be in that one). If, after the state has changed to $|u\rangle$, you measure its state again right away, you will find it to be $|u\rangle$ with probability one.

We can see that this is how polarization works. Suppose someone gives me a right-circularly polarized photon, which corresponds to is the state

$$|\circlearrowright\rangle = \frac{1}{\sqrt{2}}\big(|\uparrow\rangle + i\,|\rightarrow\rangle\big). \tag{2.4.3}$$

This tells me that if I pass the photon through a horizontally oriented sheet of polaroid, I'll find the polarization to be horizontal half the time (if it's transmitted) and vertical half the time (if it's reflected). After passing through or reflecting, the photon's polarization state has changed. But note that using the fact that

$$\begin{aligned} |\uparrow\rangle &= \frac{1}{\sqrt{2}}\big(|\nearrow\rangle + |\nwarrow\rangle\big) \\ |\rightarrow\rangle &= \frac{1}{\sqrt{2}}\big(|\nearrow\rangle - |\nwarrow\rangle\big), \end{aligned} \tag{2.4.4}$$

I can also write $|\circlearrowright\rangle$ as

$$|\circlearrowright\rangle = \frac{1}{2}\big([1+i]\,|\nearrow\rangle \;+\; [1-i]\,|\nwarrow\rangle\big). \tag{2.4.5}$$

That implies that if I confront the photon with a diagonally oriented polaroid (with P below standing for probability):

$$\begin{aligned} P(\nearrow) &= \left|\frac{1}{2}(1+i)\right|^2 = \frac{1}{4}(1+i)(1-i) = \frac{1}{2} \\ P(\nwarrow) &= \left|\frac{1}{2}(1-i)\right|^2 = \frac{1}{2}, \end{aligned} \tag{2.4.6}$$

and the state changes to $|\nearrow\rangle$ or $|\nwarrow\rangle$ during the measurement. We could easily find cases in which the probabilities are not 1/2. We'll usually be measuring the values of qubits, which corresponds to determining the probability for being in the particular basis states $|0\rangle$ (which we can take to be $|\uparrow\rangle$) and $|1\rangle$ (which we can take to be $|\rightarrow\rangle$). We will usually be discussing measurements that are with respect to this "standard" basis (as Rieffel and Polak call it) or "computational" basis (as everyone else calls it).

Note that you can't learn much about the original state $|v\rangle$ from the measurement, which gives you only one result and destroys the original state in doing so. This fact vitiates much of the hyped advantage of quantum computing — that the computer "computes all possible answers to a question at the same time." That simultaneous computation may be happening in a vague sense, but as soon as you try to check all the possible answers to see which is best, you find only one, and that one is randomly selected from the full set. And afterwards you can't get any more information about the original superposition of answers. Any useful quantum algorithm will have to be very clever to get around this problem.

Often it is easier to use the inner product explicitly than to rewrite everything in a way that allows the identification of coefficients. By noting, e.g., that we can represent right-circular polarization by

$$|\circlearrowright\rangle \doteq \frac{1}{\sqrt{2}}\begin{pmatrix}1\\ i\end{pmatrix}, \tag{2.4.7}$$

and horizontal polarization by

$$|\nwarrow\rangle \doteq \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}, \tag{2.4.8}$$

we can use the component representation of the inner product to obtain

$$\begin{aligned} P(\nwarrow) &= |\langle \nwarrow | \circlearrowleft \rangle|^2 \\ &= \left|\frac{1}{2}\begin{pmatrix}1 & -1\end{pmatrix}\begin{pmatrix}1\\i\end{pmatrix}\right|^2 \\ &= \frac{1}{4}|1-i|^2 \quad = \quad \frac{1}{2}. \end{aligned} \tag{2.4.9}$$

I could also do all that directly in the ket representation, still without rewriting the states.

Finally, it's important that the overall "phase" of the state vector is irrelevant, i.e. there is no physical difference between $|v\rangle$ and $-|v\rangle$, or more generally between $|v\rangle$ and $e^{i\varphi}|v\rangle$ for any real φ. That's because such a factor will not affect the absolute value of the dot product of the vector with any other vector. Note that this does *not* mean, however, that $a|u\rangle + b|u^\perp\rangle$ represents the same state as $a|u\rangle + be^{i\varphi}|u^\perp\rangle$. A relative phase is indeed significant, as we can see in the difference between

$$|\nearrow\rangle \equiv \frac{1}{\sqrt{2}}(|\uparrow\rangle + |\rightarrow\rangle) \tag{2.4.10}$$

and

$$|\nwarrow\rangle \equiv \frac{1}{\sqrt{2}}(|\uparrow\rangle - |\rightarrow\rangle) \tag{2.4.11}$$

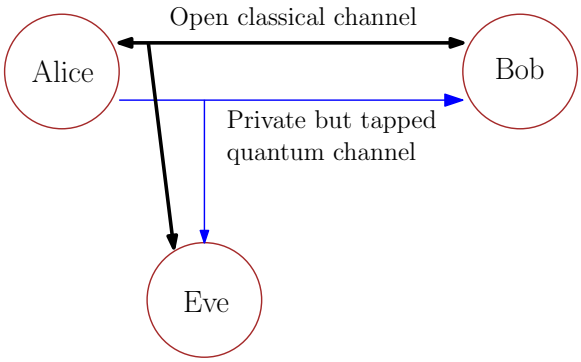
or

$$|\circlearrowleft\rangle \equiv \frac{1}{\sqrt{2}}(|\uparrow\rangle + i|\rightarrow\rangle)\,. \tag{2.4.12}$$

2.5 Quantum Key Distribution Protocol

In quantum communication discussions, the two people who want to communicate are always called Alice (A) and Bob (B). Let us suppose that these two want to exchange messages. They first need to establish a key, a random string of bits known only to the two of them. They can use the key as a "one-time code pad." Once they both have the key, Alice can send Bob a message, which will be another (non-random) string. The key can be used to encode the message by flipping every bit in the message if the corresponding bit in the key is a 1. This converts the definitely un-random codes string to another random string that anyone who has the key can convert back to the original message by flipping the appropriate bits back. If someone without the key intercepts the coded message, that person will be unable to decode it.

Bob and Alice have a completely open channel by which they can exchange normal (classical) messages, and in addition Alice can send Bob a sequence of polarized photons (the polarization of which Bob can measure) as in the schematic diagram below:



A malevolent third party, Eve, has the ability to intercept the photons Alice sends, measure their polarization (or not) and then send them back on their way to Bob. How can Alice and Bob establish a key — a sequence of 0's and 1's, which recall we represent as $\uparrow$'s and $\rightarrow$'s — so that Eve cannot learn what it is without Alice and Bob finding out? There are classical ways but, as Rieffel and Polak discuss, those are susceptible to Eve's quantum computer. Can Alice and Bob use quantum mechanics?

Here's a way that goes by the name of the BB84 protocol. Alice chooses two different encodings for her bits, the computational basis:

$$\begin{aligned} |0\rangle &\longrightarrow |\uparrow\rangle \\ |1\rangle &\longrightarrow |\rightarrow\rangle\,, \end{aligned} \tag{2.5.1}$$

and the "Hadamard basis:"

$$\begin{aligned} |0\rangle &\longrightarrow |\nearrow\rangle \\ |1\rangle &\longrightarrow |\nwarrow\rangle\,. \end{aligned} \tag{2.5.2}$$

and randomly switches between these two (and the value of the choice of polarization vectors themselves within each of these bases, say by passing the photons through a polaroid randomly oriented in the $\uparrow$, $\rightarrow$, $\nearrow$, and $\nwarrow$ directions), and sends the photons to Bob. She keeps track of the polarization of each photon she sends.

When Bob receives the photons he measures each of their polarizations, also switching randomly between the two bases. Afterwards he confirms with Alice via the open channel that he received the same number of bits that she sent, and the two tell each other which bases they used in preparing/measuring the state of each photon. They remove from consideration all cases in which they did not measure the polarization in the same basis, and then select a random subset of the remaining cases and compare bits (again over the classical channel). If they all agree, Alice and Bob use the string consisting of the bits in the remaining cases in which they measured in the same basis as the key.

How can interference by Eve be detected? Suppose Eve intercepts a photon. If she just sends it on its way to Bob without doing anything to it, she learns nothing. The only way to learn something is to measure the polarization with respect to some basis. Eve's problem is that she doesn't know ahead of time in which basis the photon polarization was prepared. If she measures it in the correct basis, she won't change anything: if, for example the photon was prepared in the state $|0\rangle$ in the computational basis, then she gets 0 with probability 1 if she measures the polarization in that basis (say by passing the

photon through a beam splitter and seeing where it comes out. She can then send the photon on to Bob without having changed its state in any way, and he will be none the wiser.

But suppose the state $|0\rangle$ was originally encoded in the Hadamard basis, so that she has received the polarization state $|\nearrow\rangle$. If Eve measures in the computational basis, she will cause the photon to change its state to either $|\uparrow\rangle$ or $|\rightarrow\rangle$, with a 50% chance for each. Then, when the photon goes to Bob, no matter which state the polarization has collapsed into, he will have a 50% chance of measuring its polarization to be $\nearrow$ and a 50% chance of measuring it to be $\nwarrow$ if he's measuring in the Hadamard basis. (If he's measuring in the computational basis, the result is discarded after the Alice-Bob basis comparison). Thus, half the time he will measure a different value for the bit than what Alice sent.

Of course, about half the time Eve will guess the right basis to measure with respect to, and Bob and Alice will agree even though Eve knows the value as well. But the other half of the time the will agree only on one out of every two or so photons. So all in all, about a quarter of the time they will disagree on the value of a bit if Eve has intercepted it. If they take a long string of bits to test on and still agree on the value of every one, they can be nearly sure that Eve has not intercepted a significant fraction. Rieffel and Polak mention various ways in which this protocol can be strengthened further, used with noisy bits, etc.

Here is a quote from Mermin's book on this protocol:

> Pause to savor this situation. Nobody has figured out how to exploit quantum mechanics to provide a secure means for directly exchanging meaningful messages. The secure exchange is possible only because the bit sequences are random. On the face of it one would think nothing could be more useless than such a transmission of noise. What is bizarre is that human ingenuity combined with human perversity has succeeded in inventing a context in which the need to hide information from a third party actually provides a purpose for such an otherwise useless exchange of random strings of bits.

2.6 Bloch Sphere

Before moving on to discuss how all what we've learned must be generalized when we're dealing with many bits, it's useful to construct a geometric picture of all the possible states a single qubit can be in. We've looked explicitly at a few, corresponding to horizontal, vertical, 45°, −45°, right-circular, and left-circular photon polarization, but there are clearly more.

The Bloch sphere is a generalization of the representation of complex number z with $|z| = 1$ by points on the unit circle. All complex numbers

$$z = x + iy \tag{2.6.1}$$

with

$$|z| = x^2 + y^2 = 1\,, \tag{2.6.2}$$

can be written, with the substitution $x = \cos\theta, y = \sin\theta$ as

$$\begin{aligned} z &= \cos\theta + i\sin\theta \\ &= e^{i\theta}\,, \end{aligned} \tag{2.6.3}$$

where the equality of the two lines in (2.6.3) is Euler's identity.

Now we want a similar representation of the qubit state

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{2.6.4}$$

with

$$|\alpha|^2 + |\beta|^2 = 1\,. \tag{2.6.5}$$

Now, letting

$$\begin{aligned} \alpha &= r_\alpha e^{i\varphi_\alpha} \\ \beta &= r_\beta e^{i\varphi_\beta}\,, \end{aligned} \tag{2.6.6}$$

we have

$$|\psi\rangle = r_\alpha e^{i\varphi_\alpha}\,|0\rangle + r_\beta e^{i\varphi_\beta}\,|1\rangle\,. \tag{2.6.7}$$

Recall now that an overall complex number with magnitude 1 in front of a state is irrelevant. Without loss of generality, as they say, we can therefore multiply $|\psi\rangle$ by $\exp(-i\varphi_\alpha)$ to get the completely general state

$$\begin{aligned} |\psi'\rangle \equiv e^{-i\varphi_\alpha}\,|\psi\rangle &= r_\alpha\,|0\rangle + r_\beta e^{i(\varphi_\beta-\varphi_\alpha)}\,|1\rangle \\ &\equiv r_\alpha\,|0\rangle + (x+iy)\,|1\rangle\,, \end{aligned} \tag{2.6.8}$$

where

$$x + iy \equiv r_\alpha e^{i(\phi_\beta-\phi_\alpha)}\,. \tag{2.6.9}$$

The normalization constraint on $|\psi'\rangle$ then implies that

$$\begin{aligned} r_\alpha^2 + |x+iy|^2 &= r_\alpha^2 + x^2 + y^2 \\ &= 1\,. \end{aligned} \tag{2.6.10}$$

The points labeled by x, y, and r_α thus not only all correspond to qubit states, but also all lie on the surface of a unit sphere.

We can go a bit further. In spherical coordinates, the usual correspondence

$$\begin{aligned} x &= r\sin\theta\cos\varphi \\ y &= r\sin\theta\sin\varphi \\ z &= r\cos\theta\,, \end{aligned} \tag{2.6.11}$$

where we're now using z to denote r_α, means that because we have a unit sphere (so that $r = 1$) we can write

$$\begin{aligned} |\psi'\rangle &= z\,|0\rangle + \big(x+iy\big)\,|1\rangle \\ &= \cos\theta\,|0\rangle + \sin\theta\,\big(\cos\varphi + i\sin\varphi\big)\,|1\rangle \\ &= \cos\theta\,|0\rangle + \sin\theta\,e^{i\varphi}\,|1\rangle\,. \end{aligned} \tag{2.6.12}$$

So we can label all single qubit states with two real parameters.

There's just one more subtlety. Our mapping of qubit states onto points on the sphere isn't quite one-to-one. The state at the north pole is given by

$$\begin{aligned} |NP\rangle &= \cos 0\,|0\rangle + \sin 0 e^{i0}\,|1\rangle \\ &= |0\rangle\,. \end{aligned} \tag{2.6.13}$$

At $\theta = \pi/2$ (with $\varphi = 0$) we have

$$\begin{aligned} |\theta = \pi/2, \varphi = 0\rangle &= \cos(\pi/2)\,|0\rangle + \sin(\pi/2) e^{i0}\,|1\rangle \\ &= |1\rangle \end{aligned} \tag{2.6.14}$$

and at the south pole,

$$\begin{aligned} |SP\rangle &= \cos\pi\,|0\rangle + \sin\pi e^{i0}\,|1\rangle \\ &= -|0\rangle\,. \end{aligned} \tag{2.6.15}$$

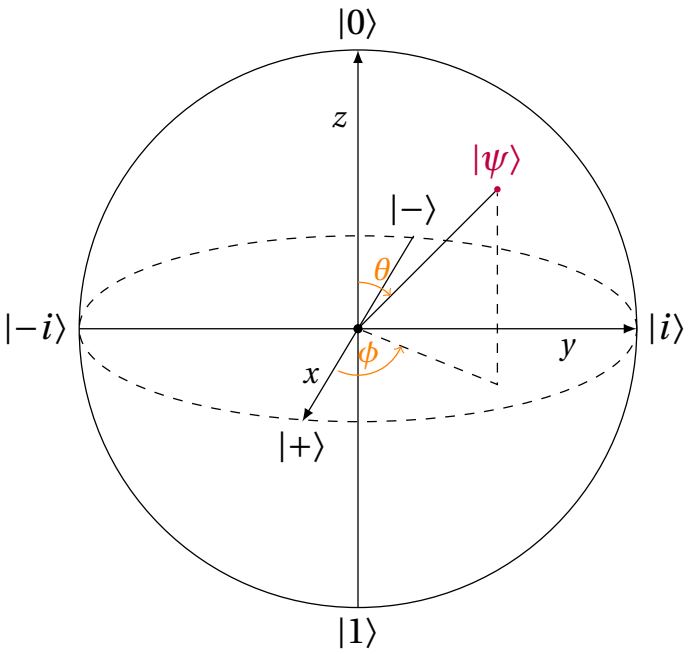
The state gets from $|0\rangle$ to $|1\rangle$ by the equator and then goes back to zero (with a minus sign, which is irrelevant) at the south pole. In fact, it's straightforward to show that states corresponding to opposite points on the sphere are the same up to a minus sign. So we get all the states from points on the upper hemisphere; the lower hemisphere repeats them.

What we need, then is to map the upper hemisphere onto the entire sphere — to stretch it out, if you will — so that each state appears only once on the sphere. It's easy to do that by replacing θ by $\theta/2$ in (2.6.12). That is, define

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + \sin\left(\frac{\theta}{2}\right) e^{i\varphi}\,|1\rangle\,. \tag{2.6.16}$$

Now the south pole corresponds to $|1\rangle$.

What about other points? Well, here's a representation (note that there's a mistake in Rieffel and Polak, at least in my version):



The labeled states corresponded to the 6 polarization states we looked at before:

$$\begin{aligned} |+\rangle &\equiv \frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big) \\ |-\rangle &\equiv \frac{1}{\sqrt{2}}\big(|0\rangle - |1\rangle\big) \\ |i\rangle &\equiv \frac{1}{\sqrt{2}}\big(|0\rangle + i\,|1\rangle\big) \\ |-i\rangle &\equiv \frac{1}{\sqrt{2}}\big(|0\rangle - i\,|1\rangle\big). \end{aligned} \tag{2.6.17}$$

3 Multi-Qubit Systems

3.1 Tensor Product

The large number of independent quantum states in a multi-qubit system comes from the way qubits combine. The usual way of combining two vector spaces to get a larger one is called the "direct sum." You can take all the vectors in the $x-y$ plane (two dimensions), for example, and add to them the vectors that point along the z axis (one dimension) to get a $2+1=3$ dimensional space. That's *not* what we'll be doing here. The "tensor product" (sometimes called the "direct product") of two spaces results in a space with a dimension equal to the dimension of the first space *times* the dimension of the second. That will mean that a computer with n qubits will have 2^n distinct quantum states.

The difference in dimension won't be so glaring with the simplest example — the tensor product of two single-qubit spaces (each with dimension 2) to get a four-qubit space — because $2+2=2\times 2=4$, The three-qubit space, however, will have dimension $4\times 2=8$, which is very different from $4+2=6$. The computational basis in the eight-dimensional space will just represent numbers from zero to seven. But back to the four-dimensional space formed by two qubits. We can construct the computational basis for the space by combining the basis states of the two one-qubit spaces. We can write these basis states as so:

$$|00\rangle \quad |01\rangle \quad |10\rangle \quad |11\rangle\,. \tag{3.1.1}$$

The meaning, e.g., of the second of these, is just that the fist qubit is in the state $|0\rangle$ *and* the second qubit is in the state $|1\rangle$. Often we write these basis states as

$$|0\rangle\,|0\rangle \qquad |0\rangle\,|1\rangle \qquad |1\rangle\,|0\rangle \qquad |1\rangle\,|1\rangle\,, \tag{3.1.2}$$

or, more formally, as

$$|0\rangle\otimes|0\rangle \qquad |0\rangle\otimes|1\rangle \qquad |1\rangle\otimes|0\rangle \qquad |1\rangle\otimes|1\rangle\,, \tag{3.1.3}$$

or, if we want to be super careful, as

$$|0\rangle_1\otimes|0\rangle_2 \qquad |0\rangle_1\otimes|1\rangle_2 \qquad |1\rangle_1\otimes|0\rangle_2 \qquad |1\rangle_1\otimes|1\rangle_2\,. \tag{3.1.4}$$

The product notation emphasizes the idea that in such states each bit is in its own one-qubit state, doing its own thing independently of the other bit.

It's important, however, these product basis states are not the only ones in the space; any linear combination of them is also a member. Thus, a generic state in the space of two qubits has the form

$$|\psi\rangle = c_1 |00\rangle + c_2 |01\rangle + c_3 |02\rangle + c_4 |10\rangle \,. \tag{3.1.5}$$

These are the superpositions of the easy-to-interpret product states. They'll be really important soon.

The math of the tensor product can of course be presented in a more formal way. Essentially, the tensor product is a way of assigning to pairs of vectors, one in an n-dimensional space and one in an m-dimensional space, a product vector in an $n \times m$-dimensional space. The assignment rule obeys

$$(|u\rangle + |v\rangle) \otimes (|w\rangle + |x\rangle) = |u\rangle \otimes |w\rangle + |u\rangle \otimes |x\rangle + |v\rangle \otimes |w\rangle + |v\rangle \otimes |x\rangle \,, \tag{3.1.6}$$

$$\begin{aligned} (c\,|u\rangle) \otimes |v\rangle &= |u\rangle \otimes (c\,|v\rangle) \\ &\equiv c\,|u\rangle \otimes |v\rangle \,, \end{aligned} \tag{3.1.7}$$

and the condition that $|u\rangle \otimes |v\rangle = |w\rangle \otimes |x\rangle$ only if $|u\rangle = c\,|w\rangle$, for some c, and $|v\rangle = \frac{1}{c}|x\rangle$ The tensor-product space contains all product vectors and their linear combinations.

Because any single qubit state is linear combination of $|0\rangle$ and $|1\rangle$, any product state of the form $|u\rangle \otimes |v\rangle$ (and any linear combination of such states) is a linear combination of the particular two-qubit states in Eq. (3.1.4), as indicated in Eq. (3.1.5). Thus these states are really a basis, as I suggested before Eq. (3.1.1).

Inner products work like this:

$$(\langle u| \otimes \langle v|)\,|w\rangle \otimes |x\rangle = \langle u|w\rangle\,\langle v|x\rangle \,. \tag{3.1.8}$$

Each part of a product state, in other words, only sees its own "kind" when it's involved an inner product. Rieffel and Polak have a long section on tensor products, while Mermin has a very short one in Appendix A. You might consult both.

3.2 Entanglement

As we said earlier, some states in a tensor product space are easy to interpret. The state

$$|\psi\rangle = |01\rangle = |0\rangle \otimes |1\rangle = |0\rangle\,|1\rangle = |0\rangle_1\,|1\rangle_2 \,,$$

where I've written the same state four ways, is just one in which the physical system that represents qubit 1 is in the state $|0\rangle$ and the system representing qubit 2 is in the state $|1\rangle$. There are similar two-qubit states that represent products of two one-quibit states that are not in the computational-basis states, e.g.

$$\begin{aligned} |\psi\rangle &= \frac{1}{\sqrt{2}}\big(|0\rangle + |1\rangle\big) \otimes \frac{1}{\sqrt{5}}\big(|0\rangle + 2i\,|1\rangle\big) \\ &= \frac{1}{\sqrt{10}}\big(|00\rangle + |10\rangle + 2i\,|01\rangle + 2i\,|11\rangle\big) \,. \end{aligned} \tag{3.2.1}$$

Although this looks complicated, it is still the case that the two qubits are co-existing in their own states.

Other states don't even have this property. Consider, for example the "Bell" state

$$|\phi\rangle = \frac{1}{\sqrt{2}}\big(|00\rangle + |11\rangle\big). \tag{3.2.2}$$

This state cannot be written in the form

$$\big(a|0\rangle + b|1\rangle\big) \otimes \big(c|0\rangle + d|1\rangle\big), \tag{3.2.3}$$

because if it could then ad and bc would both vanish, which means that two of the coefficients would be zero, while ac and bd would both equal 1, which means that none of the coefficients could be zero. Rieffel and Polak have some other examples.

The significance of a state like $|\phi\rangle$ in (3.2.2) is that it's impossible to say that qubit 1 is in one state and qubit 2 is in another. Neither qubit *has* a state of its own. If qubits are photons underneath, then neither of the two photons has its own state. There is only a two-photon state. We'll see the kind of things this fact implies when we discuss "spooky action at a distance" in a little bit.

Almost all the states in a multi-qubit system are entangled. And the question of whether a state is entangled does not depend on the basis in which you write the state. It can depend, however on how you decompose the system into subsystems. Rieffel and Polak give an example of a four-qubit state that is entangled when considered as an element of the tensor-product of four single-qubit spaces, but not when considered as an element of the product of two two-qubit spaces. In other words, the state can be written as the simple tensor product of two two-qubit states. Each of those, however, is entangled with respect to the underlying single-qubit spaces.

3.3 Measurement of Multi-Qubit States

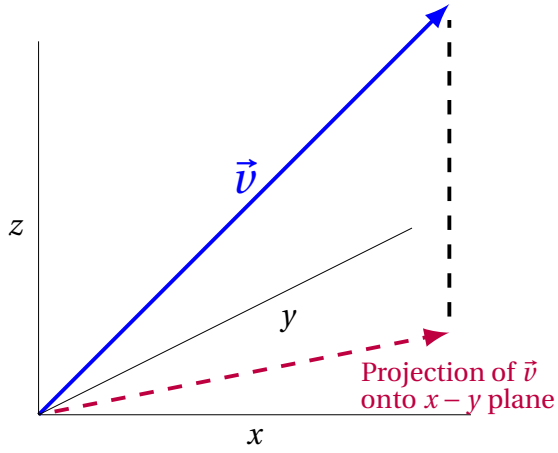
What we said about measurement for single-qubit states requires a little modification for many-qubit states. Part of the prescription is exactly the same. If, for example, you have a generic normalized two-qubit state:

$$|\Psi\rangle = a|0\rangle_1|0\rangle_2 + b|0\rangle_1|1\rangle_2 + c|1\rangle_1|0\rangle_2 + d|1\rangle_1|1\rangle_2\,, \tag{3.3.1}$$

(normalized means that $|a|^2 + |b|^2 + |c|^2 + |d|^2 = 1$), the probability of finding qubit 1 to be in state $|0\rangle$ and qubit 2 to be in state $|1\rangle$ is just $|b|^2$. But now you can do other kinds of measurements, for example measuring the state of qubit 2 while leaving qubit 1 alone. You can ask, e.g., about the probability for quibit 2 to be in the state $|0\rangle$ after such a measurement. You can't simply take the coefficient of the state $|0\rangle_2$ in $|\Psi\rangle$ because both a and c multiply terms containing $|0\rangle_2$. Which coefficient should you use?

The generalization we need (called, unsurprisingly, the "generalized Born rule," is this (and I'm using somewhat different language than the either Rieffel and Polak or Mermin do here): Instead of there being one unit vector for a given value of a quantity we want to measure, there are or can be many. Those unit vectors span a subspace of the

full tensor-product space. For a single qubit the subspace is one-dimensional — all multiples of a single vector — but it doesn't have to be for more than one qubit. To find the probability of ending up in a state in the subspace after a measurement, you "project" the state $|\Psi\rangle$ that represents the system before measurement onto the subspace. This operation is just like projecting an ordinary 3-d vector onto the xy plane; you take the component of your vector that lies in the subspace.



The squared magnitude of that component is the probability of ending up in the subspace and the piece of your original vector in the subspaces — the "projection" of the original vector onto the subspace — is the state the qubits end up in (though you have to normalize the projection so that it has length 1). It should be clear that for a single qubit this is just the Born rule; the squared magnitude of the component of a vector $|\Psi\rangle$ along a single unit vector $|u\rangle$ is just $|\langle u|\Psi\rangle|^2$.

In the example (3.3.1) above, the projection onto the two-dimensional subspace in which qubit 2 is $|0\rangle$ is

$$P|\Psi\rangle = a|0\rangle_1|0\rangle_2 + c|1\rangle_1|0\rangle_2\,, \tag{3.3.2}$$

where the symbol P above denotes the projection operation, acting on $|\Psi\rangle$. The probability for measuring qubit 2 to be in the state $|0\rangle$, i.e. the squared "length" of this projection, is thus

$$\text{Prob.} = |a|^2 + |c|^2\,, \tag{3.3.3}$$

and a *normalized* version of the state it ends up in such a case is

$$|\Psi'\rangle = \frac{1}{\sqrt{|a|^2+|c|^2}}\big(a|00\rangle + c|10\rangle\big)\,. \tag{3.3.4}$$

Note that the probability in Eq. (3.3.3) is, as it must be, just the probability of measuring qubit 2 to be in the state $|0\rangle$ *and* qubit 1 to be in the state $|0\rangle$ plus the probability of measuring qubit 20 to be in the state $|0\rangle$ *and* qubit 1 to be in the state $|1\rangle$. (That makes sense, right?)

Let's quickly look at a way of determining probabilities for measuring in different bases. We'll learn a more efficient way later, but for now we'll use brute force. Consider the state

$$|\Psi\rangle = \frac{1}{\sqrt{6}}\left(2|00\rangle + |01\rangle + |11\rangle\right)\,. \tag{3.3.5}$$

Suppose we want to know the probability of measuring the second qubit to be in the state $|+\rangle$. We can rewrite $|\Psi\rangle$ so that the qubit 2 is always in one of the Hadamard-basis states, like this:

$$\begin{aligned} |\Psi\rangle &= \frac{1}{\sqrt{6}}\left(2|0\rangle \otimes \frac{1}{\sqrt{2}}\big[\,|+\rangle + |-\rangle\,\big] + |0\rangle \otimes \frac{1}{\sqrt{2}}\big[\,|+\rangle - |-\rangle\,\big] + |1\rangle \otimes \frac{1}{\sqrt{2}}\big[\,|+\rangle - |-\rangle\,\big]\right) \\ &= \frac{1}{\sqrt{12}}\left(\big[\,3|0\rangle + |1\rangle\,\big] \otimes |+\rangle + \big[\,|0\rangle - |1\rangle\,\big] \otimes |-\rangle\right). \end{aligned} \tag{3.3.6}$$

The projection onto the subspace with qubit 2 in the state $|+\rangle$ is

$$\begin{aligned} P|\Psi\rangle &= \frac{1}{\sqrt{12}}\left(3|0\rangle + |1\rangle\right) \otimes |+\rangle \\ &= \frac{1}{\sqrt{12}}\left(3|0\rangle \otimes |+\rangle + |1\rangle \otimes |+\rangle\right), \end{aligned} \tag{3.3.7}$$

which has squared length $\frac{3^2+1^2}{12} = 5/6$. You can see from the second line of Eq. (3.3.6) that, like before, this is just the probability of measuring the second qubit to be in the state $|+\rangle$ and the first to be in the state $|0\rangle$ (which is $\frac{9}{12}$) plus the probability of measuring the second qubit to be in the state $|+\rangle$ and the first to be in the state $|1\rangle$ (which is $\frac{1}{12}$).

3.4 Using Entanglement to Distribute Keys

The following scheme is called Ekert91, after Artur Ekert. Suppose there is a source of entangled photon pairs, each in the two-qubit state

$$|\Psi\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right). \tag{3.4.1}$$

The protocol takes advantage of the fact, which I've asked you to show in the your homework, that $|\Psi\rangle$ can be rewritten as follows:

$$|\Psi\rangle = \frac{1}{\sqrt{2}}\left(|++\rangle + |--\rangle\right). \tag{3.4.2}$$

The source sends one photon from each pair to Alice and one to Bob. At some point, Alice and Bob decide to construct a key and they can do so without exchanging any photons between them. They each measure the polarization of some number of their photons, switching randomly between the computational basis and the Hadamard basis. (Bob doesn't know which basis Alice measures in, and vice versa.) Because $|\Psi\rangle$ can be written either as in (3.4.1) or (3.4.2), whenever they happen to measure in the same basis they must get the same result. Why? Well suppose Alice measures photon 1 in the computational basis and gets the result $|1\rangle$. Then the two-photon state has "collapsed" to the projection of $|\Psi\rangle$ onto the subspace in which photon 1 is in $|1\rangle$, i.e. onto the state $|11\rangle$. So when Bob measure the sate of his photon in the same basis he must also get $|1\rangle$. Similar reasoning works for the Hadamard basis and (3.4.2).

Knowing that they must have obtained the same result if they measured with respect to the same basis, Bob and Alice tell each other the basis in which they measured each

photon and discard cases in which they used different bases. For the remaining cases they each have an identical random series of results: 0, 1, +, and -. If they want they can agree to replace each + by a 0 and each - by a 1, so that they each have identical random strings of 0s and 1s. This random string is their key. If Eve tries to measure the state of one of the photons (and then forward the photon to Bob), the entangled state will collapse into one of the four product states $|00\rangle, |11\rangle, |{+}{+}\rangle, |{-}{-}\rangle$. If she measures in the same basis as Alice and Bob, she knows their bits and goes undetected. But if she measures in the wrong basis she is detected half the time, just like in the BB84 protocol. Do you see why?

In fact, Ekert91 turns out to be almost exactly the same as BB84. If Alice always measures her photon before Bob measures his, then she knows the state of Bob's photon (it's the same as hers), so it's exactly as if she sends him a photon whose polarization state she has randomly selected with a polaroid. It's just that nature is randomly choosing the state rather than Alice, in her pseudo-random way. The only practical difference is that Bob's photon arrives before Alice has to do anything, which can be really convenient. And Bob can sometimes measure first instead of Alice without affecting the protocol.

It might seem (and in fact is) strange that Alice can apparently affect the result of Bob's measurement just by conducting her own measurement on a different photon. We'll examine this situation more closely soon.

3.5 EPR

There is an something called locality that is nearly sacred in physics. It's the idea that as two objects get farther apart their influence on each other decreases. If this weren't generally the case, it would be almost impossible to do science. If what was going on in a German lab affected the behavior of objects in your lab, how could you do a controlled experiment?

There are situations in which quantum mechanics doesn't appear to respect locality, however. Einstein, Podolsky, and Rosen (EPR) realized that and used it to argue that quantum mechanics can't be a complete theory. Rather that rehearse the argument, let's just see what it is that's disturbing.

Consider the entangled two-photon polarization Bell state we just looked at:

$$|\Psi\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right). \tag{3.5.1}$$

We've seen that the photon polarizations remain entangled as the photons move apart. That implies that whenever they measure the polarization in the same basis, Alice and Bob get the same result, even even if they are a million miles apart. The reason, as we've seen, is that when one of them measures, the two-photon state collapses, requiring that the other obtain the same polarization. And that's true even if Alice measures only a tiny fraction of a second before Bob, so that a signal telling Bob's photon what Alice's had done could not reach Bob's photon before it was measured. (From all we know, no such signal can travel faster than light).

The collapse here is extremely weird. If you take quantum mechanics literally, a measurement on a photon instantaneously affects the photon thousands of miles away. Ein-

stein called this "spooky action at a distance" and didn't believe it possible. He insisted that you couldn't influence a particle by doing something to another particle very far away from it. And especially not instantaneously. So Einstein and friends would have argued that Bob's photon must have had definite horizontal polarization *before* the polarization of Alice's photon was measured (if Alice's photon was found to have horizontal polarization). Bob's photon could not somehow be given that polarization by Alice's measurement. And Bob's photon must also have had definite values for ↗ / ↖ polarization, since Alice could have caused a collapse by measuring the polarization along one of those directions as well.

Note that the quantum-mechanical description is not quite as bad as it might seem. It doesn't technically violate the principle of relativity, which says that no signal can travel faster than light, because it's not possible for Alice to tell Bob what has been measured before Bob's photon arrives. So as far as Bob is concerned, nothing weird happens; if a bunch of correlated photons are sent out in both directions, Bob sees a random sequence of horizontally and vertically polarized photons, whether Alice makes measurements or not. Thus, although there are correlations, it's hard to say what it is, if anything, that has traveled faster than the speed of light.

3.6 Hardy Paradox and Cookies

After the kerfuffle this argument raised, many people thought that perhaps quantum mechanics was incomplete, that perhaps there were "hidden variables" to which we had no access to but which determined the polarizations of the photons along all directions, and that each hidden variable always had the same value in a pair of entangled photons so that the results of polarization measurements of the two along any direction were always the same. The photons were like socks from the same pair; you could take those socks far away from each other, but any test on them would have to yield the same results because the "hidden variables" of thread quality, manufacturing process, etc. were same time for the two.

The hidden-variables for photons, whatever they happened to be, would always just reproduce the predictions of quantum mechanics, and so it didn't really matter whether you took them to exist or not. Some people thought that they could well be there but didn't worry about them much. But in the 1960's the situation changed. John Bell showed that though hidden variables can explain the correlations in experiments like those we've just discussed, they must lead to different results from those predicted by quantum mechanics in other situations, unless faster-than-light communication is possible. No replacement of quantum mechanics with the kind of theory Einstein wanted is possible! Photons cannot be like socks.

The scenario that follows, due to Lucien Hardy, illustrates this fact and is simpler than Bell's theorem. Because in the entangled quantum state we'll look at,

$$|H\rangle = \frac{1}{\sqrt{12}}\left(3\,|00\rangle + |01\rangle + |10\rangle - |11\rangle\right), \tag{3.6.1}$$

the states makeing up the entangled pair are not just $|00\rangle$ and $|11\rangle$ (that is, they're not always the same), pairs of identical socks do not make for the best ordinary-world anal-

ogy. So imagine, instead of socks, a cookie factory that places two small cookies, two at a time, on conveyor belts leading out of the factory. One cookie always goes on a belt heading east and the other on a belt heading west. The belts can be millions of miles long. At the end of each belt is is Alice (or Bob) and a testing station. Alice and Bob each make one of two tests on each cookie: popping it into her/his mouth to see whether it tastes good or bad, it or heating it up to see whether or not it steams. They can each do only one of these tests per cookie because heating may destroy a cookie's flavor and putting a cookie into your mouth adds moisture to it and could change the result of the heat test. Anyway, Alice and Bob test many cookies, randomly selecting which of the two tests to perform, and find the following results:

1. When both Alice and Bob heat their cookies, one twelfth of the time they both steam.

2. When Alice heats and finds steam and Bob tastes, his cookie always tastes good. And similarly when Bob heats and finds steam and Alice tastes, her cookie always tastes good.

Now what happens when they both taste their cookies? The above results imply, because whenever one cookie steams the other tastes good, that at least for the twelfth of the cases in which they would both steam if heated, the cookies both taste good. Right? Right.

Now we can do completely analogous tests on pairs of qubits, which you can use photons to represent. Since you can't heat or taste photons, let's make take the tests to be the measurement of polarization in the computational or Hadamard bases, and the following outcome replacements:

$$\begin{array}{lcl} \text{Cookie steams} & \longrightarrow & |1\rangle \\ \text{Cookie doesn't steam} & \longrightarrow & |0\rangle \\ \text{Cookie tastes good} & \longrightarrow & |-\rangle \\ \text{Cookie tastes bad} & \longrightarrow & |+\rangle \end{array}$$

Before the measurement the two photons are in the entangled Hardy state (3.6.1). We can see that if both Alice and Bob both measure in the computational basis, one twelfth of the time they will both get the value 1. This is just like result #1 above for the cookies.

What happens if Alice measures her qubit (the first, say) in the computational basis and Bob measures his in the Hadamard basis? Let's rewrite $|H\rangle$ to see:

$$\begin{aligned} |H\rangle = \frac{1}{\sqrt{12}}\Bigg(3\,|0\rangle \otimes \frac{1}{\sqrt{2}}\left[|+\rangle + |-\rangle\right] \;+\; |0\rangle \otimes \frac{1}{\sqrt{2}}\left[|+\rangle - |-\rangle\right] \\ + |1\rangle \otimes \frac{1}{\sqrt{2}}\left[|+\rangle + |-\rangle\right] \;-\; |1\rangle \otimes \frac{1}{\sqrt{2}}\left[|+\rangle - |-\rangle\right]\Bigg) \\ = \frac{1}{\sqrt{24}}\left(4\,|0\rangle \otimes |+\rangle + 2\,|0\rangle \otimes |-\rangle + 2\,|1\rangle \otimes |-\rangle\right). \end{aligned} \tag{3.6.2}$$

Because there is no term containing $|1\rangle \otimes |+\rangle$, whenever Alice gets the result "1," Bob must get "−". And because $|H\rangle$ is completely symmetric in the two qubits, when Bob

measures in the computational basis and Alice in the Hadamard basis, whenever Bob gets "1" Alice gets "−." This is just like result #2 above.

The state of affairs that is like both cookies tasting good occurs when Alice and Bob both measure in the Hadamard basis and both get "-". How often does that happen? Well, let's continue to rewrite $|H\rangle$, going from the last line of (3.6.2) to a representation in which both qubit states are expressed in the Hadamard basis:

$$\begin{aligned} |H\rangle &= \frac{1}{\sqrt{24}}\left(\frac{4}{\sqrt{2}}\left[|+\rangle+|-\rangle\right]\otimes|+\rangle+\frac{2}{\sqrt{2}}\left[|+\rangle+|-\rangle\right]\otimes|-\rangle+\frac{2}{\sqrt{2}}\left[|+\rangle-|-\rangle\right]\otimes|-\rangle\right) \\ &= \frac{1}{\sqrt{48}}\left(4|++\rangle+4|+-\rangle+4|-+\rangle\right). \end{aligned} \tag{3.6.3}$$

There is no term proportional to $|--\rangle$! So Alice and Bob will never both get "-" when they measure in the Hadamard basis. That's as if the cookies never both tasted good when eaten! According to quantum mechanics, photons cannot be like cookies (or socks).

In terms of qubit properties, the result is this: Suppose there were something about each photon that determined the results of any measurements on, it i.e. that photons were like cookies, which have the hidden variables of batter composition, baking temperature, etc. Then in the one-twelfth of cases in which both Bob and Alice measured in the computational basis and got "0," one of them would have gotten "−" if that person had switched to the Hadamard basis. Then if they both had switched they should both have gotten "−" in those cases. But they never both get "−" in reality. So there can't be anything about individual photons that determines the results of all experiments unless the measurement on one photon affects the properties of the other, even when no slower-than-light signal could get from one to the other before the second measurement is made. A "local" (no faster-than-light signals) hidden-variables theory cannot give the same result as quantum mechanics.

This argument does more than refute deterministic local hidden-variables theories. Even if there is randomness in the world, so that the hidden variables don't determine everything, it's still not possible for the photons to have their own variables if what happens to one photon cannot instantaneously affect the other one very far away. Suppose you try to say, for example, that one of Alice's photons state just has "the ability" to go into state $|1\rangle$ (as Mermin puts it in his Appendix C on spooky action at a distance) when measured in the computational basis, but isn't required to go into that state. Still, sometimes it *will* randomly go into that state, and if the photons are really independent of each other when far away, the variables in Bob's partner photon must then force it to go into the state $|-\rangle$ when measured in the Hadamard basis. And we can follow the same line of argument to reach the conclusion that Alice and Bob should, at least one twelfth of the time, find both photons in state $|-\rangle$ when they measure in the Hadamard basis. And, in fact, they never do. What the argument really shows is that photons are not independent entities, unless they can affect each other instantaneously at a distance.

Returning from the hypothetical world of cookies, Alice, Bob, etc., you might ask can a local hidden-variables theory describe the real world or is quantum mechanics right? The answer, experiments show, is that quantum mechanics is right. That's just the way it goes.

3.7 Linear Transformations

A transformation (or "operator"), as its name implies, is something that turns a vector into another vector. We'll sometimes represent transformations by capital letters, and the statement I just made is that, e.g.,

$$A|v\rangle = |v'\rangle \,. \tag{3.7.1}$$

Transformation play several roles in quantum mechanics but for us they will primarily represent the way the quantum computer acts on data stored as qubits. Actually, we only really need to understand *linear* transformations, which have the property

$$A(c_1|v_1\rangle + c_2|v_2\rangle) = c_1 A|v_1\rangle + c_2 A|v_2\rangle \,. \tag{3.7.2}$$

Transformations can be multiplied by numbers in an obvious associative way

$$(cA)|v\rangle = c(A|v\rangle) \equiv cA|v\rangle \,, \tag{3.7.3}$$

and the product of two transformations is similarly defined

$$(AB)|v\rangle = A(B|v\rangle) \equiv AB|v\rangle \,. \tag{3.7.4}$$

Here's an example of a linear operator: an *outer product*, denoted, e.g, by

$$|u\rangle\langle w| \tag{3.7.5}$$

where $|u\rangle$ and $|w\rangle$ are vectors. Associativity means that this can act on a ket to produce

$$|u\rangle\langle w| \;\; |v\rangle = |u\rangle\langle w|v\rangle = |u\rangle \;\; \langle w|v\rangle = (\langle w|v\rangle)\,|u\rangle \,, \tag{3.7.6}$$

where the spaces merely emphasize associativity. Rieffel and Polak use these outer products as a way to discuss arbitrary linear transformations, so we'll do the same.

Let's confine our attention for right now to single qubit (two-dimensional) systems. The generalization to many qubits is straightforward. The first thing to note is that the way a linear transformation acts on a complete set of basis vectors (two in our case) completely determines the way it acts on any vector. Why? First, because it's linear, so that (3.7.2) becomes:

$$A(c_0|0\rangle + c_1|1\rangle) = c_0 A|0\rangle + c_1 A|1\rangle \,. \tag{3.7.7}$$

Second, a sum of outer products is enough to define an arbitrary linear transformation of basis vectors. Let

$$A = a_{00}|0\rangle\langle 0| + a_{01}|0\rangle\langle 1| + a_{10}|1\rangle\langle 0| + a_{11}|1\rangle\langle 1| \,. \tag{3.7.8}$$

Then acting on the first basis vector $|0\rangle$ we get

$$\begin{aligned} A|0\rangle &= a_{00}|0\rangle \underbrace{\langle 0|0\rangle}_{1} + a_{01}|0\rangle \underbrace{\langle 1|0\rangle}_{0} + a_{10}|1\rangle \underbrace{\langle 0|0\rangle}_{1} + a_{11}|1\rangle \underbrace{\langle 1|0\rangle}_{0} \\ &= a_{00}|0\rangle + a_{10}|1\rangle \,. \end{aligned} \tag{3.7.9}$$

The last line can be any vector in the 2d space if we choose a_{00} and a_{10} properly. Similarly we can have A take $|1\rangle$ into any vector we want by properly choosing a_{10} and a_{11} in

$$\begin{aligned} A|1\rangle &= a_{00}|0\rangle \underbrace{\langle 0|1\rangle}_{0} + a_{01}|0\rangle \underbrace{\langle 1|1\rangle}_{1} + a_{10}|1\rangle \underbrace{\langle 0|1\rangle}_{0} + a_{11}|1\rangle \underbrace{\langle 1|1\rangle}_{1} \\ &= a_{01}|0\rangle + a_{11}|1\rangle \,. \end{aligned} \tag{3.7.10}$$

That means that the form (3.7.8) is general enough to take both basis vectors into any other vectors, and thus, since the action of A on basis vectors completely determines its action on any vector, is general enough to represent any linear transformation.

One of the basic ideas of linear algebra is that one can represent a linear transformation by a matrix that acts on the components of vectors to give the components of the corresponding transformed vectors. Here's how we can show with outer products: Let $|v\rangle$ be an arbitrary vector

$$|v\rangle = c_0|0\rangle + c_1|1\rangle \,. \tag{3.7.11}$$

and suppose that A takes it into another arbitrary vector $|v'\rangle$:

$$\begin{aligned} A|v\rangle &= |v'\rangle \\ &= c_0'|0\rangle + c_1'|1\rangle \,. \end{aligned} \tag{3.7.12}$$

Then, using Eqs. (3.7.9) and (3.7.10),

$$\begin{aligned} A|v\rangle &= A(c_0|0\rangle + c_1|1\rangle) \\ &= c_0 A|0\rangle + c_1 A|1\rangle \\ &= c_0(a_{00}|0\rangle + a_{10}|1\rangle) + c_1(a_{01}|0\rangle + a_{11}|1\rangle) \\ &= (a_{00}c_0 + a_{01}c_1)|0\rangle + (a_{10}c_0 + a_{11}c_1)|1\rangle \\ &= c_0'|0\rangle + c_1'|1\rangle \,, \end{aligned} \tag{3.7.13}$$

where the last line comes form (3.7.12). This implies that

$$c_0' = a_{00}c_0 + a_{01}c_1 \tag{3.7.14}$$

$$c_1' = a_{10}c_0 + a_{11}c_1 \tag{3.7.15}$$

or, in other words,

$$\begin{pmatrix} c_0' \\ c_1' \end{pmatrix} = \begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} . \tag{3.7.16}$$

We've shown that the action of an arbitrary A on an arbitrary vector, given the basis $|0\rangle, |1\rangle$, can be represented by a matrix that depends only on A (and the basis) acting on the components of the vector. If we changed the basis, both the components of the vectors and the elements of the matrix representing A would be different, though.

All this generalizes to more than two dimensions. In particular, for $N+1$ dimensions, we have — and now I'm using summation notation that it may take awhile for you to get used to (and that I'll try to avoid using too much) —

$$A = \sum_{i=0}^{N} \sum_{j=0}^{N} a_{ij} |i\rangle\langle j| \,, \tag{3.7.17}$$

where $|i\rangle$ is the i^{th} basis state, and if

$$|v\rangle = \sum_{k=0}^{N} c_k |k\rangle \,, \tag{3.7.18}$$

then, defining the "Kronecker delta"

$$\delta_{jk} = \begin{cases} 1 & j = k \\ 0 & \text{otherwise} \end{cases} \,, \tag{3.7.19}$$

we get

$$\begin{aligned} A|v\rangle &= \left(\sum_{i=0}^{N}\sum_{j=0}^{N} a_{ij}|i\rangle\langle j|\right)\left(\sum_{k=0}^{N} c_k|k\rangle\right) \\ &= \sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{k=0}^{N} a_{ij}c_k|i\rangle\underbrace{\langle j|k\rangle}_{\delta_{jk}} \\ &= \sum_{i=0}^{N}\sum_{j=0}^{N} a_{ij}c_j|i\rangle \\ &= \sum_{i=0}^{N} c'_i|i\rangle \,, \end{aligned} \tag{3.7.20}$$

where the last line comes from equation $A|v\rangle$ with $|v'\rangle$. This means that for all i,

$$c'_i = \sum_{j=0}^{N} a_{ij}c_j \,, \tag{3.7.21}$$

which is equivalent to

$$\begin{pmatrix} c'_0 \\ c'_1 \\ \vdots \\ c'_N \end{pmatrix} = \begin{pmatrix} a_{00} & a_{01} & \dots & a_{0,N} \\ a_{10} & a_{11} & \dots & a_{1,N} \\ \vdots & \vdots & \vdots & \vdots \\ a_{N0} & a_{N1} & \dots & a_{N,N} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_N \end{pmatrix} \tag{3.7.22}$$

If you don't already know that (3.7.21) is equivalent to (3.7.22), you should definitely check. That's an important fact!

We said earlier that associativity can be used to define the product of transformations:

$$AB\ |\psi\rangle \equiv A\big(B|\psi\rangle\big) \,, \qquad \forall\, |\psi\rangle \,. \tag{3.7.23}$$

Acting first with B and then with A is equivalent, with both those transformations written in the form (3.7.17), is equivalent to acting with

$$\begin{aligned} C = AB &= \sum_i\sum_j a_{ij}|i\rangle\langle j|\sum_l\sum_k b_{lk}|l\rangle\langle k| \\ &= \sum_i\sum_j\sum_k\sum_l a_{ij}b_{lk}|i\rangle\langle j|l\rangle\langle k| \\ &= \sum_i\sum_j\sum_k\sum_l a_{ij}b_{lk}|i\rangle\delta_{jl}\langle k| \\ &= \sum_i\sum_j\sum_k a_{ij}b_{jk}|i\rangle\langle k| \,, \end{aligned} \tag{3.7.24}$$

which implies that the matrix elements of the product C of two transformations A and B are given by

$$c_{ik} = \sum_j a_{ij} b_{jk}, \qquad \forall i,k. \tag{3.7.25}$$

You should verify that this is "index notation" for the matrix product:

$$\begin{pmatrix} c_{00} & c_{01} & \dots & c_{0,N} \\ c_{10} & c_{11} & \dots & c_{1,N} \\ \vdots & \vdots & \vdots & \vdots \\ c_{N0} & c_{N1} & \dots & c_{N,N} \end{pmatrix} = \begin{pmatrix} a_{00} & a_{01} & \dots & a_{0,N} \\ a_{10} & a_{11} & \dots & a_{1,N} \\ \vdots & \vdots & \vdots & \vdots \\ a_{N0} & a_{N1} & \dots & a_{N,N} \end{pmatrix} \begin{pmatrix} b_{00} & b_{01} & \dots & b_{0,N} \\ b_{10} & b_{11} & \dots & b_{1,N} \\ \vdots & \vdots & \vdots & \vdots \\ b_{N0} & b_{N1} & \dots & b_{N,N} \end{pmatrix}. \tag{3.7.26}$$

Because the matrices obey the same product relations as the transformations themselves, and, when multiplying the components of a vector give the components of the vector produced by the operator, we say that matrices "represent" the transformations.

Let's do a two-qubit example. What matrix, in the computational basis, represents the operator

$$A = 2\,|01\rangle\langle 00| + i\,|11\rangle\langle 11| - |00\rangle\langle 10|\,?$$

To answer, we need to order the computational-basis states, so we know how to put coefficients in columns. Let's choose the order $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. Then I hope you can see that

$$A \doteq \begin{pmatrix} 0 & 0 & -1 & 0 \\ 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & i \end{pmatrix}.$$

Finally for (near) future reference, we define $A^\dagger$, the adjoint (or conjugate) of the transformation A through the relation

$$\langle\psi|A^\dagger|\varphi\rangle \equiv \langle\varphi|A|\psi\rangle^* \qquad \forall\,|\psi\rangle, |\varphi\rangle\,. \tag{3.7.27}$$

The bra $\langle\psi|$ can be regarded as a "functional" (something that acts on a vector to give a number) associated with the ket $|\psi\rangle$, with the association defined by

$$\langle\psi| \quad \text{acting on} \quad |\varphi\rangle \quad \equiv \quad \langle\psi|\varphi\rangle \qquad \forall\,|\varphi\rangle\,, \tag{3.7.28}$$

If we used the dagger symbol as shorthand for this association:

$$\langle\psi| \equiv |\psi\rangle^\dagger\,, \tag{3.7.29}$$

If you use a column containing the components of $|\psi\rangle$ to represent that vector, then you represent $\langle\psi|$ by a row vector containing the complex conjugates of those components. Anyway, as I hope you can show in a few lines, it follows from Eq. (3.7.29) that

$$\langle\psi|\,A^\dagger = \langle A\psi| \equiv \big(A|\psi\rangle\big)^\dagger\,, \tag{3.7.30}$$

where the expression in the middle, $\langle A\psi|$, is a helpful (I hope) extension of Dirac notation that just means the bra corresponding to $A|\psi\rangle$ (which is exactly what $\big(A|\psi\rangle\big)^\dagger$ means). It may be homework to show that $a^\dagger_{ij}$, the elements of matrix that represents $A^\dagger$, are given by

$$a^\dagger_{ij} = a^*_{ji}\,. \tag{3.7.31}$$

3.8 Projection Operators for Measurement

I told you earlier that the generalized Born rule for calculating probabilities says that the probability of some outcome is the squared "length" of the projection of the system's state vector onto the subspace containing states corresponding to that outcome. These projections can be obtained by operating on the state-vector with "projection operators," which can in turn be represented by sums of outer products of the form $|\psi\rangle\langle\psi|$. For example, the projection operator onto the one-dimensional subspace spanned by the single-qubit state $|1\rangle$ is

$$P_1 = |1\rangle\langle 1| \,. \tag{3.8.1}$$

You can see why. If

$$|v\rangle = c_0 |0\rangle + c_1 |1\rangle \,, \tag{3.8.2}$$

then

$$\begin{aligned} P_1 |v\rangle &= |1\rangle\langle 1| \; |v\rangle \\ &= c_0 |1\rangle \langle 1|0\rangle + c_1 |1\rangle \langle 1|1\rangle \\ &= 0 + c_1 |1\rangle \times 1 \\ &= c_1 |1\rangle \,. \end{aligned} \tag{3.8.3}$$

The result is just the piece of $|v\rangle$ that lies in the space spanned by $|1\rangle$, as it should be.

Suppose I want to project a vector onto the subspace of two-qubit space in which both bits have the same value (say, to determine whether they do). The full two-qubit space has four dimensions and the subspace I'm interested in is that space spanned by $|00\rangle$ and $|11\rangle$. Note that the projection of any vector onto this space will be the sum of its projection onto $|00\rangle$ and its projection onto $|11\rangle$. I could draw a picture to demonstrate that, but it's easy to see by considering an arbitrary two-qubit state:

$$|\Psi\rangle = a|00\rangle + b|01\rangle + c|10\rangle + d|11\rangle \,, \tag{3.8.4}$$

The projection onto the subspace in which the two qubits have equal values is

$$P_{\text{equal}} |\Psi\rangle = a|00\rangle + d|11\rangle \,. \tag{3.8.5}$$

I hope it's clear that this is the sum of the projection onto the $|00\rangle$ projection onto $|11\rangle$. That means that the projector can be written in the form

$$P_{\text{equal}} = |00\rangle\langle 00| + |11\rangle\langle 11| \,. \tag{3.8.6}$$

This formalism is more useful when you project onto a space that is not so easy to represent in the computational basis. How would you compute the probability of measuring the first bit of a state

$$|\psi\rangle = \frac{1}{\sqrt{3}} \left(|00\rangle - |10\rangle + |11\rangle\right) \tag{3.8.7}$$

to be $|+\rangle$. Well, the projector onto the subspace in which qubit 1 is $|+\rangle$ can be written as

$$P_{q_1=+} = |+0\rangle\langle +0| + |+1\rangle\langle +1| \tag{3.8.8}$$

because it projects onto all possible subspaces with qubit 1 in the state $|+\rangle$. Applying $P_{q_1=+}$ to $|\psi\rangle$ gives

$$\begin{aligned} P_{q_1=+}|\psi\rangle = \frac{1}{\sqrt{3}}\Bigg(& |+0\rangle \underbrace{\langle +0|00\rangle}_{1/\sqrt{2}} + |+1\rangle \cancel{\langle +1|00\rangle} - \underbrace{|+0\rangle\langle +0|10\rangle}_{1/\sqrt{2}} - |+1\rangle \cancel{\langle +1|10\rangle} \\ & + |+0\rangle \cancel{\langle +0|11\rangle} + \underbrace{|+1\rangle\langle +1|11\rangle}_{1/\sqrt{2}} \Bigg) \\ = \frac{1}{\sqrt{6}}(|+0\rangle - |+0\rangle + |+1\rangle) \\ = \frac{1}{\sqrt{6}}|+1\rangle\,. \end{aligned} \tag{3.8.9}$$

Note that the state of the first bit is $|+\rangle$, as it must be. The probability is then just the squared magnitude of this projection:

$$\text{Prob}_{q_1=+} = \frac{1}{6}. \tag{3.8.10}$$

4 Transforming Quantum States

4.1 Unitary Transformations

We've seen that linear transformations turn quantum states into other quantum states. In quantum mechanics, all changes to quantum states that occur when a system is *not* being measured are represented by special linear transformations, ones that are *unitary*. Such transformations are those that "preserve inner products between vectors," so that probabilities always add up to 1. This means that the inner product of $U|a\rangle$ with $U|b\rangle$ is the same as the inner product of $|a\rangle$ with $|b\rangle$, i.e., from Eq. (3.7.30),

$$\begin{aligned} \langle a|\,U^\dagger \quad U|b\rangle &= \langle a|U^\dagger U|b\rangle \\ &= \langle a|b\rangle\,, \quad \forall\ |a\rangle, |b\rangle \end{aligned}$$

or

$$U^\dagger U = 1$$

or

$$U^{-1} = U^\dagger$$

or

$$UU^\dagger = 1\,.$$

The product of two unitary transformations is unitary, and so is the tensor product.

Since $U^\dagger$ is defined for any transformation U, so is U^{-1}, which means that all unitary transformations are "reversible." This in turn implies that any quantum computation must be reversible. Most classical computers are constructed in a way that is not

reversible. It's possible, however, to show that any classical computation can be implemented in a reversible way without a great loss of efficiency. We're not going to do that, but will look at a few examples of how to implement traditional computational operations (such as addition) with quantum circuits.

Just to get you a little familiar with unitary transformations, let's quickly derive a general form for the matrices that represent one-qubit transformations. A general 2×2 matrix can be written as

$$U = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \tag{4.1.1}$$

and so it takes 8 real parameters to specify such a matrix. The unitarity condition restricts that:

$$\begin{aligned} U^\dagger &= \begin{pmatrix} a^* & c^* \\ b^* & d^* \end{pmatrix} \\ UU^\dagger &= \begin{pmatrix} aa^* + bb^* & ac^* + bd^* \\ ca^* + db^* & cc^* + dd^* \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \end{aligned} \tag{4.1.2}$$

which implies

$$|a|^2 + |b|^2 = 1 \qquad |c|^2 + |d|^2 = 1 \qquad ac^* + bd^* = 0\,. \tag{4.1.3}$$

The last of these conditions can be satisfied by writing

$$\begin{aligned} c &= Ae^{i\varphi} b^* \quad \text{completely general, real } A \text{ and } \varphi \\ \Longrightarrow d &= -Ae^{i\varphi} a^* \end{aligned} \tag{4.1.4}$$

Since $|a|^2 + |b|^2 = |c|^2 + |d|^2$,

$$\begin{aligned} |a|^2 + |b|^2 &= A^2|b^2| + A^2|a|^2 \\ \Longrightarrow A &= 1 \end{aligned} \tag{4.1.5}$$

and

$$U = \begin{pmatrix} a & b \\ e^{i\varphi} b^* & -e^{i\varphi} a^*\,. \end{pmatrix} \tag{4.1.6}$$

It takes five real parameters to determine a matrix of the form (4.1.6) but the requirement that $|a|^2 + |b|^2 = 1$ means that only four parameters can be specified independently. Soon we'll look at a systematic way choosing four parameters and writing the transformations in terms of them.

4.2 No-Cloning Theorem

The fact that all quantum state changes correspond to linear transformations has an important consequence: there is no way to "clone" an arbitrary quantum state. A cloning machine would correspond to a transformation U that takes a state $|v\rangle\,|0\rangle$ into

the state $|v\rangle|v\rangle$ for all states $|v\rangle$. Consider three states $|u\rangle$, $|v\rangle$, and $|w\rangle \equiv a|u\rangle + b|v\rangle$. The cloning operator would have to do the following:

$$\begin{aligned} U|u\rangle|0\rangle &= |u\rangle|u\rangle \\ U|v\rangle|0\rangle &= |v\rangle|v\rangle \\ U|w\rangle|0\rangle &= |w\rangle|w\rangle \\ &= (a|u\rangle + b|v\rangle)(a|u\rangle + b|v\rangle)\,. \end{aligned} \tag{4.2.1}$$

But because U is linear, if the top two lines of Eq. (4.2.1) are true, then

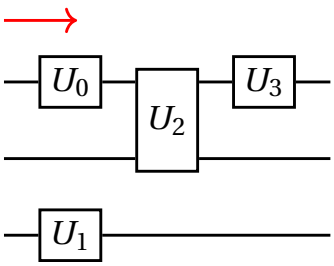
$$\begin{aligned} U|w\rangle|0\rangle &= U(a|u\rangle + b|v\rangle)|0\rangle \\ &= U(a|u\rangle|0\rangle + b|v\rangle|0\rangle) \\ &= a|u\rangle|u\rangle + b|v\rangle|v\rangle \\ &\neq |w\rangle|w\rangle\,. \end{aligned} \tag{4.2.2}$$

So if, in other words, the machine successfully clones $|u\rangle$ and $|v\rangle$, it won't clone $|w\rangle$. That means, for example, that Eve cannot copy the photon polarization state she receives from Alice in the BB84 protocol and send the copy to Bob while analyzing Alice's photon for herself.

4.3 Gates

We will adopt the "circuit model" for describing quantum computing algorithms. Circuits (or more precisely, "gate arrays") consist of a sequence of gates that act on one or more bits at at time. A gate is simply a unitary transformation of the qubit states that enter it. The term "model" is used, I imagine, because there are no circuits in the usual sense. The terminology is borrowed from classical computing but doesn't imply that anything is actually flowing anywhere. In most physical realizations, nothing really moves around.

Here is an example of a simple gate array:



The three "wires" represent single-qubit states and the boxes represent unitary transformations that act on those states. The "flow" is from left to right, as the arrow indicates. First U_0 acts on the first qubit and U_1 on the third. Then U_2 acts simultaneously on the (altered) first qubit and second qubit, and finally U_3 acts on the first qubit.

4.4 Simple One-Qubit Gates

The simplest gates act on only one qubit at a time. The simplest of these is the identity transformation, which does nothing:

$$\begin{aligned} I &= |0\rangle\langle 0| + |1\rangle\langle 1| \\ I|0\rangle &= |0\rangle \\ I|1\rangle &= |1\rangle\,. \end{aligned} \tag{4.4.1}$$

In any basis, I is represented by the identity matrix

$$I \doteq \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{4.4.2}$$

Next simplest are the "Pauli" gates X, Y, and Z, which in the physics of electron spin (the simplest two-state system) are denoted $\sigma_x, i\sigma_y$, and σ_z. The X gate, in the computational basis, is like a NOT. It transforms the basis states as follows:

$$\begin{aligned} X &= |0\rangle\langle 1| + |1\rangle\langle 0| \\ X|0\rangle &= |1\rangle \\ X|1\rangle &= |0\rangle\,. \end{aligned} \tag{4.4.3}$$

You should be able to see that in the computational basis the transformation X is represented by

$$X \doteq \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{4.4.4}$$

The next simplest is is Z, which flips the sign of the coefficient of $|1\rangle$ while leaving $|0\rangle$ alone:

$$\begin{aligned} Z &= |0\rangle\langle 0| - |1\rangle\langle 1| \\ Z|0\rangle &= |0\rangle \\ Z|1\rangle &= -|1\rangle\,. \end{aligned} \tag{4.4.5}$$

This transformation is represented (again, as you can check) by the matrix

$$Z \doteq \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{4.4.6}$$

The last of these is Y:

$$\begin{aligned} Y &= |0\rangle\langle 1| - |1\rangle\langle 0| \\ Y|0\rangle &= -|1\rangle \\ Y|1\rangle &= |0\rangle\,, \end{aligned} \tag{4.4.7}$$

the matrix representation for which is

$$Y \doteq \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}. \tag{4.4.8}$$

Y is often defined elsewhere with an additional factor of $-i$, so that it equals σ_y (for those of you who've taken a QM class).

Any linear transformation of a single-qubit state can be written as a linear combination of the four transformations I, X, Y, and Z. Perhaps showing that will appear on your homework?

One such linear combination, and incredibly useful one, is the Hadamard transformation H:

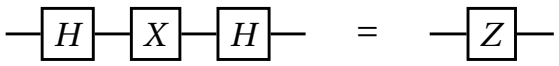
$$H = \frac{1}{\sqrt{2}}\big(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|\big)$$
$$H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{4.4.9}$$
$$H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)\,.$$

The matrix that represents H in the computational basis is

$$H \doteq \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{4.4.10}$$

Let's try one of the problems on the homework.

Prove the following:



Let's first do this the long way with explicit expressions for the operators:

$$\begin{aligned} HXH &= \frac{1}{\sqrt{2}}(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|) \times (|0\rangle\langle 1| + |1\rangle\langle 0|) \\ &\quad \times \frac{1}{\sqrt{2}}(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 1| - |1\rangle\langle 0|) \\ &= \frac{1}{\sqrt{2}}(|0\rangle\langle 1| + |0\rangle\langle 0| + |1\rangle\langle 1| - |1\rangle\langle 0|) \times \frac{1}{\sqrt{2}}(|0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1|) \\ &= \frac{1}{2}(|0\rangle\langle 0| - |0\rangle\langle 1| + |0\rangle\langle 0| + |0\rangle\langle 1| + |1\rangle\langle 0| - |1\rangle\langle 1| - |1\rangle\langle 0| - |1\rangle\langle 1|) \\ &= |0\rangle\langle 0| - |1\rangle\langle 1| \\ &= Z\,. \end{aligned} \tag{4.4.11}$$

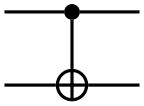
It's easier in matrix form (in the computational basis):

$$\begin{aligned} HXH &\doteq \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\ &= \frac{1}{2}\begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\ &= \frac{1}{2}\begin{pmatrix} 2 & 0 \\ 0 & -2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \\ &\doteq Z \end{aligned} \tag{4.4.12}$$

We could also have simply looked at what happens to $|0\rangle$ and $|1\rangle$, as I did in class. The action of a transformation on a set of basis vectors determines its action on any vector, so showing that HXH has the same action as Z on set of basis is enough to show that the two transformations are identical.

4.5 Two-Qubit Gates: Controlled Operations

Among gates that act on two qubits at a time, the most common are those that in the computational basis act on one of the qubits in one way if the other bit is in the state $|0\rangle$, and another way if it's in the state $|1\rangle$. Here is the graphical representation of a "controlled-NOT" gate:



There are other ways of drawing this operation, though this one (perhaps the least clear at first) is the most common. The "control" bit is the one with the small solid circle and the "target" bit, which is sometimes flipped, by the one with the circled plus sign. We can write the transformation down explicitly in terms of outer products:

$$C_{\text{not}} = |00\rangle\langle 00| + |01\rangle\langle 01| + |11\rangle\langle 10| + |10\rangle\langle 11| \,, \tag{4.5.1}$$

so that

$$\begin{aligned} C_{\text{not}}\,|00\rangle &= |00\rangle \\ C_{\text{not}}\,|01\rangle &= |01\rangle \\ C_{\text{not}}\,|10\rangle &= |11\rangle \\ C_{\text{not}}\,|11\rangle &= |10\rangle \end{aligned} \tag{4.5.2}$$

If we want to we can also write this in matrix form, provided we choose a basis (which must be ordered). For the computational basis, ordered $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, we get, as I hope you can check for yourself,

$$C_{\text{not}} \doteq \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{4.5.3}$$

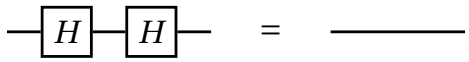
The reason for the circle plus sign in the diagram is that we can also represent (4.5.2) in a compact way:

$$C_{\text{not}}\,|x\,y\rangle = |x, x \oplus y\rangle \,, \tag{4.5.4}$$

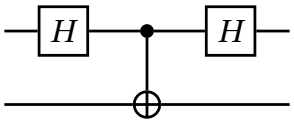
where the symbol $\oplus$ means addition modulo 2, which is equivalent to the "exclusive or" (denoted XOR):

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

It's important to realize that this kind of gate doesn't always act the way your intuition would suggest. The following identity is easy to prove:



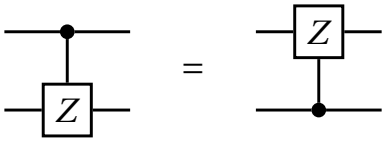
But consider the following gate array:



Because the controlled-NOT in the middle might seem to alter only the state of the bit on the bottom, and because $HH = I$, you might think that this array will never change the top bit. But here's where entanglement rears its head. Let's look at what the array does to $|00\rangle$, where the first bit is on top:

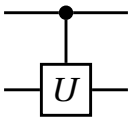
$$\begin{aligned} |00\rangle &\xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle_1 + |1\rangle_1)\,|0\rangle_2 \;=\; \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle) \\ &\xrightarrow{C_{\text{not}}} \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\ &\xrightarrow{H} \frac{1}{\sqrt{2}}\left[\frac{1}{\sqrt{2}}(|0\rangle_1 + |1\rangle_1)\,|0\rangle_2 + \frac{1}{\sqrt{2}}(|0\rangle_1 - |1\rangle_1)\,|1\rangle_2\right] \\ &= \frac{1}{2}(|00\rangle + |10\rangle + |01\rangle - |11\rangle)\,. \end{aligned} \tag{4.5.5}$$

You end up with a superposition in which bit 1 has been flipped in some terms, even though it is the "control." This is related to the action of the cNOT in the second line of (4.5.5); it entangles the two previously unentangled bits. Either later in class or in your homework, you will see the somewhat weird result:



where the diagram contains "controlled Z" gates. This is more straightforward notation than that we're using for the controlled NOT; if we were to use the same notation everywhere, we'd draw the controlled NOT as a control bit connected to an X box.

These controlled gates can be generalized:



Here the single-qubit transformation U is applied to the second bit if the first bit is, in the computational basis, in the state $|1\rangle$. Rieffel and Polak use the notation $\bigwedge U$ for this

kind of gate. The example it looks at is the "controlled phase shift" $\bigwedge e^{i\theta}$. The gate has the form:

$$\bigwedge e^{i\theta} = |00\rangle\langle 00| + |01\rangle\langle 01| + e^{i\theta}|10\rangle\langle 10| + e^{i\theta}|11\rangle\langle 11|\,,$$

so that it doesn't change the state if the control bit is zero but multiplies it by a phase if the control bit is one. Note that although the operator is "officially" acting on only one of the qubits (with the action controlled by the other) its effect when it acts is to change the two-qubit state by a phase, which doesn't distinguish between the control and target qubits. The controlled phase shift doesn't change the physical content of any of the computational states (it at worst adds a global phase), but acting on the superposition

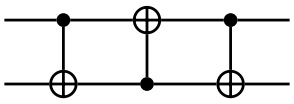
$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right),$$

it gives

$$\bigwedge e^{i\theta}|\psi\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + e^{i\theta}|11\rangle\right),$$

which contains only a local phase and thus represents a different state than does $|\psi\rangle$.

You can construct useful circuits from controlled gates. Consider the following one:



What does it do? Well

$$\begin{aligned}
|00\rangle &\to |00\rangle \to |00\rangle \to |00\rangle \\
|01\rangle &\to |01\rangle \to |11\rangle \to |10\rangle \\
|10\rangle &\to |11\rangle \to |01\rangle \to |01\rangle \\
|11\rangle &\to |10\rangle \to |10\rangle \to |11\rangle\,.
\end{aligned}$$

Thus the circuit simply swaps the two qubits in the computational basis. Actually, it does this in any basis:

$$\begin{aligned}
|\psi\rangle|\varphi\rangle &\to (a|0\rangle_1 + b|1\rangle_1)(c|0\rangle_2 + d|1\rangle_2) \\
&= ac|00\rangle + ad|01\rangle + bc|10\rangle + bd|11\rangle \\
&\longrightarrow ac|00\rangle + ad|10\rangle + bc|01\rangle + bd|11\rangle \\
&= ca|00\rangle + cb|01\rangle + da|10\rangle + db|11\rangle \\
&= (c|0\rangle + d|1\rangle) \otimes (a|0\rangle + b|1\rangle) \\
&= |\varphi\rangle|\psi\rangle\,. \quad \checkmark
\end{aligned}$$

4.6 Dense Encoding and Teleportation

At this point we can understand how, by applying single-qubit transformations to her own qubit in a shared entangled pair, Alice can

1. send two bits-worth of information
2. "teleport" an unknown qubit quantum state.

Let's look at first at dense encoding. Alice and Bob each have just a single qubit, but Alice can encode (and Bob can then decode) all numbers between 0 and 3, i.e. the amount of information in two qubits. The basic idea here is that by manipulating just one bit of an entangled pair, Alice is able to create four different orthogonal Bell states. I don't know who thought of the protocol, but here's how it goes:

Alice and Bob are each sent one qubit of the usual entangled EPR pair

$$|\psi_0\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) .$$

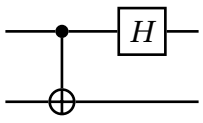
Alice acts on her qubit (the first one). Surprisingly, she can put the entangled pair into any of the four orthogonal Bell states by manipulating only her qubit. To do that, she transforms it in one of the following ways, depending on which number she wants to encode:

$$\begin{aligned}
\text{Encoding } 0: \quad & |\psi_0\rangle \to I \otimes I\,|\psi_0\rangle \\
& = |\psi_0\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) \\
\text{Encoding } 1: \quad & |\psi_0\rangle \to X \otimes I\,|\psi_0\rangle \\
& = |\psi_1\rangle = \frac{1}{\sqrt{2}}\left(|10\rangle + |01\rangle\right) \\
\text{Encoding } 2: \quad & |\psi_0\rangle \to Z \otimes I\,|\psi_0\rangle \\
& = |\psi_2\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle - |11\rangle\right) \\
\text{Encoding } 3: \quad & |\psi_0\rangle \to Y \otimes I\,|\psi_0\rangle \\
& = |\psi_3\rangle = \frac{1}{\sqrt{2}}\left(-|10\rangle + |01\rangle\right)
\end{aligned}$$

Alice then sends her qubit to Bob so that he can use both to decode. If Bob is able to carry out a measurement in the Bell basis, that's all he has to do. But if he isn't he can reverse the encoding with a couple of unitary transformations on the pair. Noting that

$$\begin{aligned}
H|+\rangle &= \frac{1}{\sqrt{2}}\left(|+\rangle + |-\rangle\right) = |0\rangle \\
H|-\rangle &= \frac{1}{\sqrt{2}}\left(|+\rangle - |-\rangle\right) = |1\rangle ,
\end{aligned}$$

and recalling that a combination of H and C_{not} gate entangles qubits and can therefore also disentangle them (C_{not} is its own inverse, as is H), Bob then runs the pair through the following circuit:



and the following happens (depending on what Alice has done):

$$
\begin{aligned}
|\psi_0\rangle &\xrightarrow{C_{\text{not}}} \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |0\rangle \\
&\xrightarrow{H\otimes I} |00\rangle \\
|\psi_1\rangle &\xrightarrow{C_{\text{not}}} \frac{1}{\sqrt{2}}(|11\rangle + |01\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |1\rangle \\
&\xrightarrow{H\otimes I} |01\rangle \\
|\psi_2\rangle &\xrightarrow{C_{\text{not}}} \frac{1}{\sqrt{2}}(|00\rangle - |10\rangle) = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes |0\rangle \\
&\xrightarrow{H\otimes I} |10\rangle \\
|\psi_3\rangle &\xrightarrow{C_{\text{not}}} \frac{1}{\sqrt{2}}(-|11\rangle + |01\rangle) = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \otimes |1\rangle \\
&\xrightarrow{H\otimes I} |11\rangle
\end{aligned}
$$

Cool!

You might have noticed, despite the coolness, that although Alice only sends two bits of information with only one qubit, Alice and Bob together need two qubits to make this scheme work. So what good will it be? One use might be in networks that are busy sometimes and not at others. During the slow periods you use the network to deliver entangled pairs. Then, when things get busy, you use those pairs to double the speed at which messages can be sent.

The idea behind teleportation is closely related the one behind dense encoding. Suppose Alice has a bit in an unknown state that she wants to transmit to Bob. She can't clone it, but she can "teleport" it, i.e. allow Bob to construct it while destroying her own copy, providing that Alice and Bob share the same two-qubit state as before. Let Alice's own (additional) bit be in the state

$$|\varphi\rangle = a|0\rangle + b|1\rangle\ ,$$

so that the total three-qubit state is

$$
\begin{aligned}
|\psi_0\rangle &= |\varphi\rangle \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \\
&= a|000\rangle + a|011\rangle + b|100\rangle + b|111\rangle\ .
\end{aligned}
$$

Alice has control over the first two qubits and runs them through the same circuit that Bob used above. That circuit entangles the unentangled states:

$$
\begin{aligned}
|00\rangle &\xrightarrow{C_{\text{not}}} |00\rangle \quad \xrightarrow{H\otimes I} \quad \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle) \\
|01\rangle &\xrightarrow{C_{\text{not}}} |01\rangle \quad \xrightarrow{H\otimes I} \quad \frac{1}{\sqrt{2}}(|01\rangle + |11\rangle) \\
|10\rangle &\xrightarrow{C_{\text{not}}} |11\rangle \quad \xrightarrow{H\otimes I} \quad \frac{1}{\sqrt{2}}(|01\rangle - |11\rangle) \\
|11\rangle &\xrightarrow{C_{\text{not}}} |10\rangle \quad \xrightarrow{H\otimes I} \quad \frac{1}{\sqrt{2}}(|00\rangle - |10\rangle)\ ,
\end{aligned}
$$

so that the three-qubit state becomes

$$\begin{aligned}|\psi\rangle &= \frac{1}{\sqrt{2}}\left[a\left(|000\rangle + |100\rangle\right) + a\left(|011\rangle + |111\rangle\right)\right.\\ &\qquad \left. + b\left(|010\rangle - |110\rangle\right) + b\left(|001\rangle - |101\rangle\right)\right]\\ &= \frac{1}{\sqrt{2}}\left[|00\rangle\left(a|0\rangle + b|1\rangle\right) + |01\rangle\left(a|1\rangle + b|0\rangle\right)\right.\\ &\qquad \left. + |10\rangle\left(a|0\rangle - b|1\rangle\right) + |11\rangle\left(a|1\rangle - b|0\rangle\right)\right] .\end{aligned}$$

Alice then measures the first two bits and tells the results to Bob, who then applies a transformation to the third qubit (the one he has access to) that depends on the outcome of Alice's measurement in the following way:

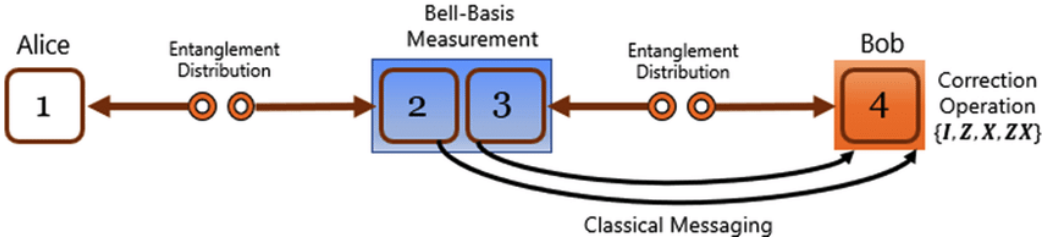
Receives	Applies
00	I
01	X
10	Z
11	Y

These are just the same transformations Alice applied when creating the dense coding. When Bob applies the appropriate transformation to the state he receives, he gets

$$\begin{aligned}I\frac{1}{\sqrt{2}}\left(a|0\rangle + b|1\rangle\right) &= \frac{1}{\sqrt{2}}\left(a|0\rangle + b|1\rangle\right)\\ X\frac{1}{\sqrt{2}}\left(a|1\rangle + b|0\rangle\right) &= \frac{1}{\sqrt{2}}\left(a|0\rangle + b|1\rangle\right)\\ Z\frac{1}{\sqrt{2}}\left(a|0\rangle - b|1\rangle\right) &= \frac{1}{\sqrt{2}}\left(a|0\rangle + b|1\rangle\right)\\ Y\frac{1}{\sqrt{2}}\left(a|1\rangle - b|0\rangle\right) &= \frac{1}{\sqrt{2}}\left(a|0\rangle + b|1\rangle\right) .\end{aligned}$$

So Bob always gets Alice's original state out!

Something very much like teleportation, called entanglement swapping, can be used to establish entanglement between photons that are very far away, even if the entanglement can only be maintained over short distances. The idea, as illustrated below, is that two people located at different places between Alice and Bob each prepare entangled pairs, and each send one of their photons (labeled 1 and 4 in the figure) to Alice or Bob and the other (labeled 2 and 3) to a lab halfway between Alice and Bob that is close enough so that the entanglement is not degraded.



Someone in the lab does a measurement in the Bell basis, and the communicates the result to Bob. Bob is then able to carry out operations, just as in the teleportation protocol, that lead to photons 1 and 4 being entangled. If there is another lab and pair source between Bob and someone else (Carol?) then photon 1 created by Alice can become entangled with one of Carol's photons even farther away. A setup that uses many such entanglement swaps is called a "quantum repeater." The details of entanglement swapping and an experiment to establish it are described in a comprehensible way (I hope) in Phys. Rev. Lett. **80**, 3981 (1998), available for now, at least, at http://web.physics.ucsb.edu/~quopt/swap.pdf

Rieffel and Polak point out that teleportation and dense coding are sort of like inverses to one another because when Alice encodes the state to teleport she uses the same transformations that Bob uses to decode the dense encoding, and when Alice does the dense encoding she uses the same transformations that Bob uses to isolate the teleported bit.

4.7 Most General Single-Qubit Unitary Transformation

Every quantum state, up to a global phase, corresponds to a point on the Bloch sphere. We want to find the most general unitary transformation that maps points on the sphere to to other points on the sphere. Mermin and Rieffel and Polak present rigorous but rather tedious derivations of the form of such a transformation; here is a more intuitive discussion.

Recall that the state corresponding to the coordinates θ, φ on the sphere have the form

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + \sin\left(\frac{\theta}{2}\right)e^{i\varphi}|1\rangle\ . \tag{4.7.1}$$

Recall also that a linear transformation is completely specified by its action on a pair of basis states. And a transformation U is unitary if it maps two orthonormal basis states onto two other orthonormal basis states. The reason for that last fact is that unitary transformations are those that preserve inner products between any two vectors, and the inner products between two arbitrary vectors

$$\begin{aligned} |\psi\rangle &= a|0\rangle + b|1\rangle \\ |\varphi\rangle &= c|0\rangle + d|1\rangle\ , \end{aligned} \tag{4.7.2}$$

and their mapped counterparts

$$\begin{aligned} U|\psi\rangle &= a\,U|0\rangle + b\,U|1\rangle \\ U|\varphi\rangle &= c\,U|0\rangle + d\,U|1\rangle\ , \end{aligned} \tag{4.7.3}$$

are the same as long as $U|0\rangle$ and $U|1\rangle$ are orthonormal. The reason for *that* is that in an orthonormal basis, the inner products depend only on the coefficients a, b, c, d. The consequence of all this for us is that we only have to find the most general transformation that maps $|0\rangle$ and $|1\rangle$ onto a pair of orthogonal states on the Bloch sphere, i.e. onto states at opposite points on the sphere. In other words, the most general unitary transformation is one that maps $|0\rangle$ to an arbitrary point on the sphere and $|1\rangle$ to a point

opposite it. "Most general" means that it can also give arbitrary global phases to the two points. We can expect such a transformation to depend on four parameters, two to specify the polar coordinates (of, say the state that $|0\rangle$ transforms into) on the Bloch sphere and two for the global phases attached to $|0\rangle$ and $|1\rangle$.

So how do we explicitly write down a transformation that takes the north pole to an arbitrary point θ, φ, while mapping the south pole to $\pi - \theta, \varphi + \pi$ and adding phases?. The standard way is as follows: First we apply a transformation that gives one phase γ to $|0\rangle$ and the opposite phase to $|1\rangle$:

$$\begin{aligned} T(\gamma)|0\rangle &= e^{i\gamma}|0\rangle \\ T(\gamma)|1\rangle &= e^{-i\gamma}|1\rangle\,. \end{aligned} \tag{4.7.4}$$

This transformation clearly preserves the orthonormality of the two states. Then we make a transformation that corresponds to rotating the points that represent our states on the Bloch sphere by and angle 2β (you'll understand the factor of 2 when you do your homework) around the y axis:

$$\begin{aligned} R(\beta)T(\gamma)|0\rangle &= e^{i\gamma}\left(\cos\beta|0\rangle - \sin\beta|1\rangle\right) \\ R(\beta)T(\gamma)|1\rangle &= e^{-i\gamma}\left(\sin\beta|0\rangle + \cos\beta|1\rangle\right). \end{aligned} \tag{4.7.5}$$

This also clearly preserves orthonormality (you can check if you want) Then we apply T again, with a different argument α. At non-zero θ, which we now have because of the rotation around the y axis of the Bloch sphere by β, $T(\alpha)$ corresponds to a rotation by 2α around the z axis.

$$\begin{aligned} T(\alpha)R(\beta)T(\gamma)|0\rangle &= e^{i\gamma}\left(e^{i\alpha}\cos\beta|0\rangle - e^{-i\alpha}\sin\beta|1\rangle\right) \\ &= e^{i(\gamma+\alpha)}\left(\cos\beta|0\rangle + e^{i(\pi-2\alpha)}\sin\beta|1\rangle\right) \\ T(\alpha)R(\beta)T(\gamma)|1\rangle &= e^{-i\gamma}\left(e^{i\alpha}\sin\beta|0\rangle + e^{-i\alpha}\cos\beta|1\rangle\right) \\ \beta|1\rangle & \\ &= e^{-i(\gamma-\alpha)}\left(\sin\beta|0\rangle + e^{-2i\alpha}\cos\beta|1\rangle\right). \end{aligned} \tag{4.7.6}$$

The transformed states must still be orthogonal (and you can check that they are) because T is unitary. The choice $\beta = \theta/2, \alpha = (\pi - \varphi)/2, \gamma = -\alpha = (\varphi - \pi)/2$ gives the form (4.7.1) for the state that $|0\rangle$ transforms into; setting $\gamma \neq -\alpha$ allows it to gain an additional global phase, with the state that $|1\rangle$ transforms into getting a related phase. Finally, an overall phase change of both states by an additional amount δ, generated by the operator

$$K(\delta) = e^{i\delta} I \tag{4.7.7}$$

allows the global phases of the two states to be independent (because there are two parameters, γ and δ that can be adjusted to fix them). So the most general unitary transformation, including arbitrary global phases, can be represented in the form

$$U = K(\delta)T(\alpha)R(\beta)T(\gamma)\,, \tag{4.7.8}$$

with the four parameters $\alpha, \beta, \gamma, \delta$ arbitrary. There is no point in applying the last global phase change $K(\delta)$ to a single bit in isolation, but in a controlled single-bit transformation it will affect the two-qubit state if the control bit is initially in a superposition of $|0\rangle$ and $|1\rangle$.

4.8 Most General Controlled Single-Qubit Transformation

Suppose we have a single-qubit transformation

$$Q = K(\delta)\,T(\alpha)\,R(\beta)\,T(\gamma)\,, \tag{4.8.1}$$

for some $\alpha, \beta, \gamma, \delta$. (We've just shown that any single-qubit transformation can be expressed in this form.) It's possible to implement a controlled version of the transformation, which Rieffel and Polak denote by $\bigwedge Q$, only in terms of single-qubit transformations and C_{not}'s. How? Well, for starters, we note that we can "factor" this:

$$\bigwedge Q = \bigwedge K(\delta) \bigwedge Q' \tag{4.8.2}$$

with

$$Q' = T(\alpha)\,R(\beta)\,T(\gamma)\,. \tag{4.8.3}$$

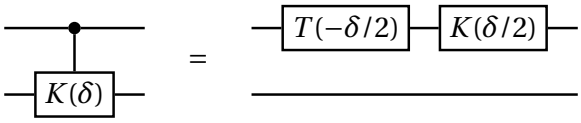
Why? Well,

$$\begin{aligned} \bigwedge K(\delta) \bigwedge Q' &= \big(|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes K(\delta)\big)\big(|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes Q'\big) \\ &= |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes K(\delta)Q' \\ &= \bigwedge\big(K(\delta)Q'\big) \\ &= \bigwedge Q \quad \checkmark \end{aligned} \tag{4.8.4}$$

Here's how you implement the first factor in terms of single-bit gates:

$$\begin{aligned} \bigwedge K(\delta) &= |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes K(\delta) \\ &= |0\rangle\langle 0| \otimes I + e^{i\delta}\,|1\rangle\langle 1| \otimes I \\ &= \Big(|0\rangle\langle 0| + e^{i\delta}\,|1\rangle\langle 1|\Big) \otimes I \\ &= e^{i\delta/2}\Big(e^{-i\delta/2}\,|0\rangle\langle 0| + e^{i\delta/2}\,|1\rangle\langle 1|\Big) \otimes I \\ &= K(\frac{\delta}{2})\,T(-\frac{\delta}{2}) \otimes I\,. \end{aligned} \tag{4.8.5}$$

This relation has the following diagrammatic form:



You might verify that this is correct by looking at what what each side of the diagrammatic equation does to states of the form $|0\rangle\,|\psi\rangle$ and $|1\rangle\,|\psi\rangle$.

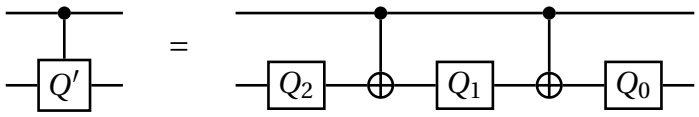
The controlled-Q' is a little more complicated. The idea is to combine pieces of Q' with C_{not}'s so that any transformations on the "target bit" are undone when the C_{not} doesn't do anything. One way to do that is to define three operators

$$\begin{aligned} Q_0 &= T(\alpha)R(\beta/2) \\ Q_1 &= R(-\beta/2)\,T\left(\frac{-\gamma-\alpha}{2}\right) \\ Q_2 &= T\left(\frac{\gamma-\alpha}{2}\right). \end{aligned} \tag{4.8.6}$$

Then we claim that

$$\bigwedge Q' = (I\otimes Q_0)\,C_{\text{not}}\,(I\otimes Q_1)\,C_{\text{not}}\,(I\otimes Q_2)\,, \tag{4.8.7}$$

or, in diagrammatic form:



Why? Well, if the top bit is in the state $|0\rangle$, then the bottom bit in state $|x\rangle$ goes to

$$\begin{aligned} |x\rangle &\longrightarrow Q_0Q_1Q_2\,|x\rangle \\ &= T(\alpha)R(\beta/2)R(-\beta/2)T\left(\frac{-\gamma-\alpha}{2}\right)T\left(\frac{\gamma-\alpha}{2}\right)|x\rangle \\ &= T(\alpha)T(-\alpha)\,|x\rangle \\ &= |x\rangle\,, \end{aligned} \tag{4.8.8}$$

while if the top bit is in the state $|1\rangle$, then the bottom bit goes to

$$\begin{aligned} |x\rangle &\longrightarrow Q_0XQ_1XQ_2\,|x\rangle \\ &= T(\alpha)R(\beta/2)\;\;XR(-\beta/2)X\;\;XT\left(\frac{-\gamma-\alpha}{2}\right)X\;\;T\left(\frac{\gamma-\alpha}{2}\right)|x\rangle\,, \end{aligned} \tag{4.8.9}$$

where I've inserted $XX\equiv I$ between the R and T operators in Q_1. What does surrounding an operator with X on both sides do? In class I may have examined the question by using outer products to write operators, but here I'll use matrix representations:

$$\begin{aligned} XAX &\doteq \begin{pmatrix}0&1\\1&0\end{pmatrix}\begin{pmatrix}a_{00}&a_{01}\\a_{10}&a_{11}\end{pmatrix}\begin{pmatrix}0&1\\1&0\end{pmatrix} \\ &= \begin{pmatrix}a_{10}&a_{11}\\a_{00}&a_{01}\end{pmatrix}\begin{pmatrix}0&1\\1&0\end{pmatrix} \\ &= \begin{pmatrix}a_{11}&a_{10}\\a_{01}&a_{00}\end{pmatrix}. \end{aligned} \tag{4.8.10}$$

The matrix represented by $R(\beta)$ is

$$R(\beta) \doteq \begin{pmatrix}\cos\beta&\sin\beta\\-\sin\beta&\cos\beta\end{pmatrix}, \tag{4.8.11}$$

so $XR(\beta)X$ is represented by

$$\begin{aligned} XR(\beta)X &\doteq \begin{pmatrix} \cos\beta & -\sin\beta \\ \sin\beta & \cos\beta \end{pmatrix} \\ &\doteq R(-\beta)\,. \end{aligned} \tag{4.8.12}$$

Similarly, $XT(\alpha)X = T(-\alpha)$. So

$$\begin{aligned} T(\alpha)R(\beta/2)\;\; XR(-\beta/2)X\;\; XT\left(\frac{-\gamma-\alpha}{2}\right)X\;\; T\left(\frac{\gamma-\alpha}{2}\right) & \\ = T(\alpha)R(\beta/2)R(\beta/2)T\left(\frac{\gamma+\alpha}{2}\right)T\left(\frac{\gamma-\alpha}{2}\right) & \\ = T(\alpha)R(\beta)T(\gamma) & \\ = Q'\,, & \end{aligned} \tag{4.8.13}$$

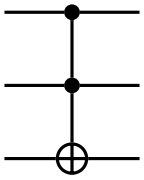
and

$$|x\rangle \longrightarrow Q'|x\rangle\,, \tag{4.8.14}$$

when the top bit is 1. That's what we set out to show.

4.9 Multiply Controlled Single-Qubit Transformations

It's possible to use more than one qubit to control the action of a gate. Consider the "controlled-controlled-NOT' gate, also called the Toffoli gate:



This gate, which Rieffel and Polak denote $\bigwedge_2 X$ (where the subscript indicates the number of controlling bits) acts as a "not" on the lowest qubit if and only if the two upper qubits are in the state $|1\rangle$. Just as we can write the action of the ordinary C_{not} transformation in the form

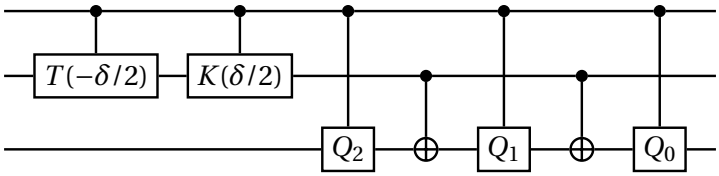
$$C_{\text{not}}|x,y\rangle = \bigwedge X|x,y\rangle = |x, x\oplus y\rangle\,, \tag{4.9.1}$$

we can write the action of the Toffoli gate as

$$T \equiv \bigwedge\nolimits_2 |x,y,z\rangle = |x,y,z\oplus xy\rangle\,. \tag{4.9.2}$$

We'll see shortly that a quantum circuit that can perform any classical computation can be built solely out of Toffoli gates.

Any multiply-controlled gate can be built out of one- and two-qubit gates. Consider the doubly controlled Q, where Q is the completely general single-qubit gate we discussed in the last section. The double controlled Q can be constructed as in this figure:



This circuit does nothing if the top bit is zero. If it is, and if the middle bit is also zero, that is obvious. If the middle bit is one (while the top one is zero) the only things that act are two NOTs on the bottom bit; they just cancel each other. If, however, the top bit is one, the effect on the bottom two lines is exactly the same as in the singly-controlled-Q circuit. So our circuit really is a doubly-controlled-Q.

If there are more control bits, the procedure we just laid out to construct a doubly-controlled gate from a singly-controlled one can just be repeated.

4.10 General n-qubit Transformations

We didn't cover this topic, and you're not responsible for it, but just in case you're curious, I include notes on it. Here we show that any unitary transformation U on n qubits can be constructed from single-qubit and C_{not} gates. First we'll show that any unitary transformations can be expressed as a sequence of "2×2" unitary transformations — transformations which act only on a two-dimensional subspace while leaving the rest alone. Then we'll show how to write any 2×2 unitary transformation in terms of single-qubit transformations and C_{not}'s.

First, construct a basis for the $N = 2^n$ states: $|\varphi_1\rangle, |\varphi_2\rangle, \ldots, |\varphi_N\rangle$. Then consider the action of our arbitrary N-bit unitary transformation U on the first basis state $|\varphi_1\rangle$:

$$U|\varphi_1\rangle = a_1|\varphi_1\rangle + a_2|\varphi_2\rangle + a_3|\varphi_3\rangle + \cdots + a_N|\varphi_N\rangle\,. \tag{4.10.1}$$

We can write this as $W_1|\varphi 1\rangle$, where $W_1 = W_1^{(a)}W_1^{(b)}\ldots$ is a product of $N-1$ unitary transformations that act as follows:

$$\begin{aligned}
W_1^{(a)}:&\quad |\varphi_1\rangle \longrightarrow a_1|\varphi_1\rangle + b_1|\varphi_2\rangle\\
W_1^{(b)}:&\quad b_1|\varphi_2\rangle \longrightarrow a_2|\varphi_2\rangle + b_2|\varphi_3\rangle\\
W_1^{(c)}:&\quad b_2|\varphi_3\rangle \longrightarrow a_3|\varphi_3\rangle + b_3|\varphi_4\rangle\\
&\quad\vdots\\
W_0^{(N-1)}:&\quad b_{N-2}|\varphi_{N-1}\rangle \longrightarrow a_{N-1}|N-1\rangle + a_N|\varphi_N\rangle\,,
\end{aligned} \tag{4.10.2}$$

where the b_i are chosen to preserve the make sure that the mapped states are normalized (so that the transformation is unitary). Each line above corresponds to the action of a different 2×2 unitary transformation. The first, $W_1^{(a)}$, acts only in the subspace spanned by $|\varphi_1\rangle$ and $|\varphi_2\rangle$ (leaving everything else alone), the second only in the subspace spanned by $|\varphi_2\rangle$ and $|\varphi_3\rangle$. Thus, although W_1, the product of all these 2×2 unitary transformations, is different from U, it does the same thing as U to $|\varphi_1\rangle$. Now define $U_1 = W_1^{-1}U$ and note that

$$\begin{aligned}
U_1|\varphi_1\rangle = W_1^{-1}U|\varphi_1\rangle &= W_1^{-1}W_1|\varphi_1\rangle\\
&= |\varphi_1\rangle\,.
\end{aligned} \tag{4.10.3}$$

So U_1 leaves $|\varphi_1\rangle$ alone and alters only the subspace spanned by $\{|\varphi_2\rangle, |\varphi_3\rangle, \ldots, |\varphi_N\rangle\}$.

Now we can repeat the construction, defining W_2 as a product of 2×2 unitary transformations such that $W_2|\varphi_1\rangle = |\varphi_1\rangle$ and $W_2|\varphi_2\rangle = U_1|\varphi_2\rangle$, then defining $U_2 = W_2^{-1}U_1$ so that U_2 preserves both $|\varphi_1\rangle$ and $|\varphi_2\rangle$. Continuing like that, we eventually get

$$W_{N-1}^{-1}W_{N-2}^{-1}\ldots W_1^{-1}U = I\,, \tag{4.10.4}$$

or, in other words

$$U = W_1 W_2 \ldots W_{N-1}\,, \tag{4.10.5}$$

a product of products (i.e. a product) of 2×2 unitary transformations.

The next part is to show that any 2×2 unitary transformation can be written in terms of single-qubit transformations and C_{not}'s. This may seem obvious because the 2×2 transformations we've encountered act on only a single qubit. But that isn't true for all 2×2 transformations; they can transform any two-dimensional computational-basis subspace of the full vector space (leaving the rest of it alone), not just the subspace corresponding to changing a single bit. Let's look at an example to understand this. Consider the three-qubit transformation:

$$\begin{aligned} |000\rangle &\longrightarrow a|000\rangle + b|111\rangle \\ |111\rangle &\longrightarrow c|000\rangle + d|111\rangle \\ \text{All orthogonal states} &\longrightarrow \text{themselves}\,. \end{aligned} \tag{4.10.6}$$

In the computational basis with the usual ordering, this transformation (which we call U) has the matrix representation

$$U \doteq \begin{pmatrix} a & 0 & 0 & 0 & 0 & 0 & 0 & c \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ b & 0 & 0 & 0 & 0 & 0 & 0 & d \end{pmatrix}. \tag{4.10.7}$$

The 2×2 transformation $\tilde{U}$ that acts in the subspace spanned by $|000\rangle$ and $|111\rangle$ has the representation (in that subspace)

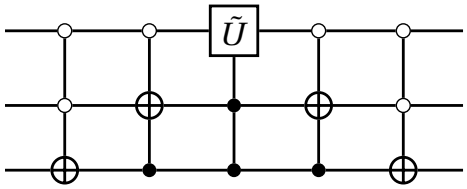
$$\tilde{U} \doteq \begin{pmatrix} a & c \\ b & d \end{pmatrix}. \tag{4.10.8}$$

The idea for implementing U is to swap bits one-at a time between $|000\rangle$ and the other basis states until the state that was originally $|000\rangle$ differs in only one qubit from $|111\rangle$. Then we apply the transformation $\tilde{U}$ as a one-qubit transformation between those two states. Finally, we swap back. Let's see how this works explicitly.

First we need what is called a Gray code, a sequence of strings that differ by only one bit and interpolate between the two states that $\tilde{U}$ mixes, in this case $|000\rangle$ and $|111\rangle$. The Gray code we want here (the shortest) is

Num.	String
1	000
2	001
3	011
4	111

When we've swapped two qubits according to this code, so that $|000\rangle$ has become $|011\rangle$, we can apply $\tilde{U}$ to the first qubit. The circuit that does the swapping, implements the 2×2 transformation, and finally swaps back is this:



We can trace through what happens to some of the input strings.

$$
\begin{aligned}
|000\rangle &\to |001\rangle \to |011\rangle \to a|011\rangle + b|111\rangle \\
&\to a|001\rangle + b|111\rangle \to a|000\rangle + b|111\rangle \\
|111\rangle &\to |111\rangle \to |111\rangle \to c|011\rangle + d|111\rangle \\
&\to c|001\rangle + d|111\rangle \to c|000\rangle + d|111\rangle|001\rangle \quad \to |000\rangle \to |000\rangle \to |000\rangle \\
&\to |000\rangle \to |001\rangle \\
|011\rangle &\to |011\rangle \to |001\rangle \to |001\rangle \\
&\to |011\rangle \to |011\rangle \,.
\end{aligned}
\tag{4.10.9}
$$

You can check for yourselves that nothing happens to the other four inputs. When everything's all over, the two states we wanted to transform have been transformed, and the others have been left alone, all with a single-qubit transformation and some C_{not}'s. This procedure obviously generalizes to more qubits, and so, because we've also seen that we can write any N-qubit transformation into a sequence of 2×2 transformations, we can build a circuit that implements any unitary transformation from single-qubit gates and C_{not}'s.

The last part of chapter 5 in Rieffel and Polak show that any single-qubit transformation can be approximated to arbitrary accuracy by the repeated application of two or three such transformations. I'm going to skip the demonstration, which is a generalization of the demonstration that you can end up arbitrarily close to any point on the circle by repeatedly increasing your angle around it by the same irrational number of degrees. I will also skip an advanced topic that neither of our main texts covers — the Solovay-Kitaev theorem — which states that the approximation can be done more "efficiently."

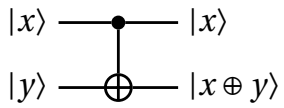
5 Quantum Algorithms: Simple Examples

5.1 Classical Computing with Qubits

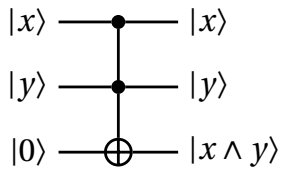
Although not all linear transformations have inverses, all unitary transformations do. That means that any quantum circuit must be "reversible," i.e. that another circuit can be built that takes the output from the first circuit and reproduces the input, that is that runs the first circuit backwards. No information, in other words, is lost. The computers we use now constantly destroy information. Any time two bits are run through an AND gate, the two input bits are replace by a single output bit, from which one can't reconstruct the input. If the output is 0, for example, we can't tell whether the two input bits were both 0 or whether one was 1.

Physically, the destruction of information leads to the generation of heat. The fastest supercomputers use huge amounts of power. Some people think that a significant improvement in computing will require reversible computation, which in principle can require very little power. We're not talking about quantum computation here, just reversible implementations of classical logic gates. The bulk of Rieffel and Polak's chapter 6 is devoted to showing that one can do that without a significant loss in computational efficiency. The implication is that quantum computers as well as reversible classical computers can carry out any classical computation with comparable efficiency to what we use now.

I'm not going to cover that, but instead will just talk a little about how to implement classical Boolean logic with quantum gates. Because the computations must be reversible, we will always have the same number of input and output gates. We've already seen that the X gate implements NOT, and that the C_{not} gate implements XOR as follows:



The Toffoli gate $T \equiv \bigwedge_2 X$ can implement AND:



or, in equation form

$$T\,|x, y, 0\rangle = |x, y, x \wedge y\rangle\ . \tag{5.1.1}$$

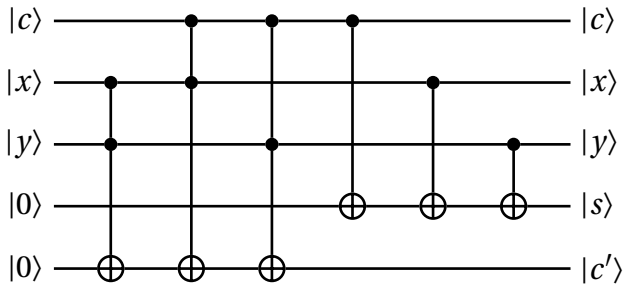
It's not hard to show that the Toffoli gate can also implement NOT, XOR, and NAND:

$$\begin{aligned} T\,|1, 1, x\rangle &= |1, 1, \neg x\rangle \\ T\,|1, x, y\rangle &= |1, x, x \oplus y\rangle \\ T\,|x, y, 1\rangle &= |x, y, \neg(x \wedge y)\rangle\ . \end{aligned} \tag{5.1.2}$$

These relations imply that the Toffoli gate by itself is enough to implement a quantum version of any classical computation, though the implementation wouldn't be very efficient.

Rieffel and Polak mention another three-qubit gate — a controlled swap known as the Fredkin gate — that can also implement a complete set of Boolean gates.

Finally, here is a piece of an "adding" circuit that does something more complicated: it takes two binary digits, one at some position in the middle of a string of bits and the other in the same position in the middle of a second string, and puts the sum of the two bits in $|s\rangle$, taking into account the bit that was carried from the previous two-bit sum (which is in $|c\rangle$) and computing the bit to be carried over into the next two-bit sum, which is put in $|c'\rangle$. (Even the description is complicated.)

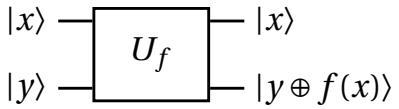


How does this work? Well, the bit that is carried over (c') should be a 1 if at least two of the bits x, y, and c are 1's. The first three Toffoli gates flip the carry bit if the their control bits are both 1. Thus if x, y, and c are all 0, or if only one of them is 1, none of the Toffoli gates will flip the carry bit, which will therefore stay at $c' = 0$. If two of the variables x, y, and c are 1, then just one of the Toffoli gates will cause a flip, and the output will be $c' = 1$. And if all three are 1, then all three Toffoli gates flip the output bit, which still make $c' = 1$. That is the behavior we want.

The new value s of the bit itself is the mod-2 sum of x, y, and c. The three C_{nots} cause s to flip each time one of them is 1. That is how mod-2 addition works. Each time you add 0 the mod-2 sum stays the same and each time you add 1 if changes 0 to 1 or 1 to 0. So the circuit indeed works as advertised. To add many-bit numbers, you can string a bunch of these one-bit adders together, with the bottom wire on the right from one circuit running to the top on the left of the next one. There are more efficient ways to do addition, but this is probably the most straightforward.

Rieffel and Polak concoct their own language for circuits and go on to describe many more classical operations. The only one that will be of real interest to us is modular exponentiation, which we will need for Shor's factoring algorithm. We'll tackle that later.

Finally, suppose we want a circuit that evaluates a generic function. As in all the examples above, we usually cannot "over-write" input bits with output. You might think that we can, that we can have an input qubit $|x\rangle$ going to an output qubit $|f(x)\rangle$. But we can only do that if f is one to one, that is, if $f(0) \neq f(1)$. Otherwise, from $f(x)$ alone we cannot reconstruct x, so the transformation is not reversible. Instead we have to have separate input and output groups of bits ("registers"), neither of which we ever discard. Suppose f maps n-digit numbers onto m-digit numbers. Then the transformation U_f that evaluates the function on the n-qubit state $|x\rangle$ works like this:



This means that each bit of y gets flipped if and only if the corresponding bit of $f(x)$ is a 1, so that you can easily extract $f(x)$ from the output. If $y = 0$, in fact, the output state is just $|f(x)\rangle$. You can see that the controlled-not/XOR is exactly this type of transformation, for one bit going to one bit. What function does $f(x)$ does it correspond to? Right, $f(x) = x$.

The point of this kind of construction is to make U_f unitary. But is it? Yes, it is, because

$$\langle x, y|x', y'\rangle = \langle x, y \oplus f(x)|x', y' \oplus f(x')\rangle \,, \tag{5.1.3}$$

so that inner products are preserved. The reason (5.1.3) holds is that if $x \neq x'$ then both sides of the equation are zero, and if $x = x'$ then $f(x) = f(x')$ and thus the two sides are either both zero again (if $y \neq y'$) or both one (if $y = y'$).

Quantum computing takes advantage of the possibility of superposing lots of different classical numbers. Suppose we want to evaluate a function, as in the circuit in the previous diagram, simultaneously for a whole bunch of different numbers. Let's assume that $|x\rangle$ is an n-qubit state and suppose that the n qubits are initially in the state $|000\ldots0\rangle$. If we run this initial state through $H \otimes H \otimes \ldots \otimes H$ (one H for each qubit in x) we get the state

$$\begin{aligned} |\Psi_{\text{input}}\rangle &= \frac{1}{\sqrt{2^n}} (|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes \cdots \otimes (|0\rangle + |1\rangle) \\ &= \frac{1}{\sqrt{2^n}} (|00\ldots00\rangle + |00\ldots01\rangle + |00\ldots10\rangle + |00\ldots11\rangle + \cdots + |11\ldots11\rangle) \\ &= \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle \,, \end{aligned} \tag{5.1.4}$$

where "input" means the part that doesn't include the bit or bits in the initial state $|y\rangle$ that lead to the output in the diagram above. $|\Psi_{\text{input}}\rangle$ is an equally weighted superposition of states containing every number between 0 and $2^n - 1$. If there is only one output bit y, then running the superposition together and output bit with $y = 0$ through the circuit gives

$$U_f |\Psi_{\text{in}}\rangle |0\rangle = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle |f(x)\rangle \,, \tag{5.1.5}$$

a superposition in which every possible number x is matched with the $f(x)$ (or $1 \oplus f(x)$ if $y = 1$).

This sounds great, but the problem is that you have to measure something to get any information from the state. Measuring what's in the input register after the action of U_f in Eq. (5.1.5) will cause the state to collapse into $|x\rangle |f(x)\rangle$ for some random value of x over which you have no control. Rieffel and Polak give a cute example of this. Consider a Toffoli gate with input qubits x_1 and x_2 and output qubit y, the initial value of which is 0. We know that this gate implements the transformation

$$T\,|x_1 x_2\rangle\,|0\rangle = |x_1 x_2\rangle\,|x_1 \wedge x_2\rangle\ . \tag{5.1.6}$$

If we run $|00\rangle\,|0\rangle$ through $H \otimes H \otimes I$ before U_f we get

$$U_f\,(H \otimes H \otimes I)\,|00\rangle\,|0\rangle = \frac{1}{2}\sum_{x_1,x_2=0}^{1} |x_1 x_2\rangle\,|x_1 \wedge x_2\rangle\ . \tag{5.1.7}$$

The final state contains all four rows of an "AND" table. But if we measure the input bits, we will get a completely random row. We will learn the AND of two numbers without having any control over what those numbers are. Algorithms have to be cleverly designed to extract useful information from superpositions.

We'll start next with a simple example of such cleverness.

5.2 Deutsch Problem

There are four possible one-bit-to-one-bit functions:

	$f(0)$	$f(1)$
f_0	0	0
f_1	0	1
f_2	1	0
f_3	1	1

Suppose that we are given a "black box," (called an "oracle" in the quantum-computing world) that evaluates one of these functions and we want to know which function it is. The only way to do that with a classical computer is to call the function twice to obtain $f(0)$ and $f(1)$. It's the same with a quantum computer; superposition doesn't help because even though we can create the state

$$|\Psi\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle\,|f(0)\rangle + |1\rangle\,|f(1)\rangle\right), \tag{5.2.1}$$

we still have to measure something to get the value of the function. If we measure the input bit, we will only get the function at one value of the input, and we can't even specify which input!

It turns out, though, that if we are satisfied with more limited information, quantum mechanics can provide a speedup. Suppose we ask only whether the function is constant, i.e. whether it is either f_0 or f_3, which each map 0 and 1 to the same value. With a classical computer we'd still have to ask the f-oracle twice ($f(0)$ or $f(1)$ by itself can't tell you about a property that reflects the value of the function at both values). But it turns out that with a quantum computer you only have to query the oracle, which runs the transformation U_f that represents f, only one time. It might seem like that this factor of two speedup is trivial because the problem is so simple, but in fact f could be quite complicated. It might, for example, be defined by the billionth bit of the binary representation of $\sqrt{1+x}$. This f would the billionth bit of $\sqrt{2}$ when the input is 0 and of $\sqrt{3}$

when the input is 1. We might be interested in whether these two bits have the same values. Calling the function once instead of twice might represent a significant savings.

Before getting into where the savings come from, let's see how the oracle could implement U_f for the case in which one of the straightforward functions in the table above is represented. Recall that such a transformation must have separate input and output qubits and take form

$$U_f |x\rangle |y\rangle = |x\rangle |y \oplus f(x)\rangle \tag{5.2.2}$$

If the output qubit is in the initial state $|0\rangle$, this becomes

$$U_f |x\rangle |0\rangle = |x\rangle |f(x)\rangle \tag{5.2.3}$$

With this initial value for y, the f's can be represented like this:

f	U_f
f_0	$I \otimes I$
f_1	C_{not}
f_2	$(I \otimes X)\, C_{\text{not}}$
f_3	$I \otimes X$

So someone could "easily" build an oracle that would implement U_f for some f that we don't know.

OK, so we have the oracle and we want to know whether its f is a constant, but we only want to call U_f once. How do we proceed? Well, starting from the state $|00\rangle$ (the two qubits are the input and output) the first step is to create a superposition of both input and output, with some signs thrown in:

$$\begin{aligned}(H \otimes H)(I \otimes X)|00\rangle &= (H \otimes H)|01\rangle \\ &= \frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle - |1\rangle) \\ &= \frac{1}{2}(|00\rangle - |11\rangle + |10\rangle - |01\rangle)\,.\end{aligned} \tag{5.2.4}$$

U_f acting on this then gives

$$\begin{aligned}U_f\,(H \otimes H)\,(I \otimes X)\,|00\rangle = \frac{1}{2}\big(&|0\rangle\,|f(0)\rangle - |1\rangle\,|1 \oplus f(1)\rangle \\ &+ |1\rangle\,|f(1)\rangle - |0\rangle\,|1 \oplus f(0)\rangle\big)\,.\end{aligned} \tag{5.2.5}$$

If $f(0) = f(1)$ this is

$$\begin{aligned}&\frac{1}{2}(|0\rangle + |1\rangle)\big(|f(0)\rangle - |1 \oplus f(0)\rangle\big) \\ &= \frac{1}{\sqrt{2}}|+\rangle\big(|f(0)\rangle - |1 \oplus f(0)\rangle\big)\,,\end{aligned} \tag{5.2.6}$$

while if $f(0) = 1 \oplus f(1)$, it's

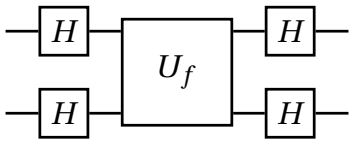
$$\begin{aligned}&\frac{1}{2}(|0\rangle - |1\rangle)\big(|f(0)\rangle - |1 \oplus f(0)\rangle\big) \\ &= \frac{1}{\sqrt{2}}|-\rangle\big(|f(0)\rangle - |1 \oplus f(0)\rangle\big)\,,\end{aligned} \tag{5.2.7}$$

Now, applying H to the input qubit gives

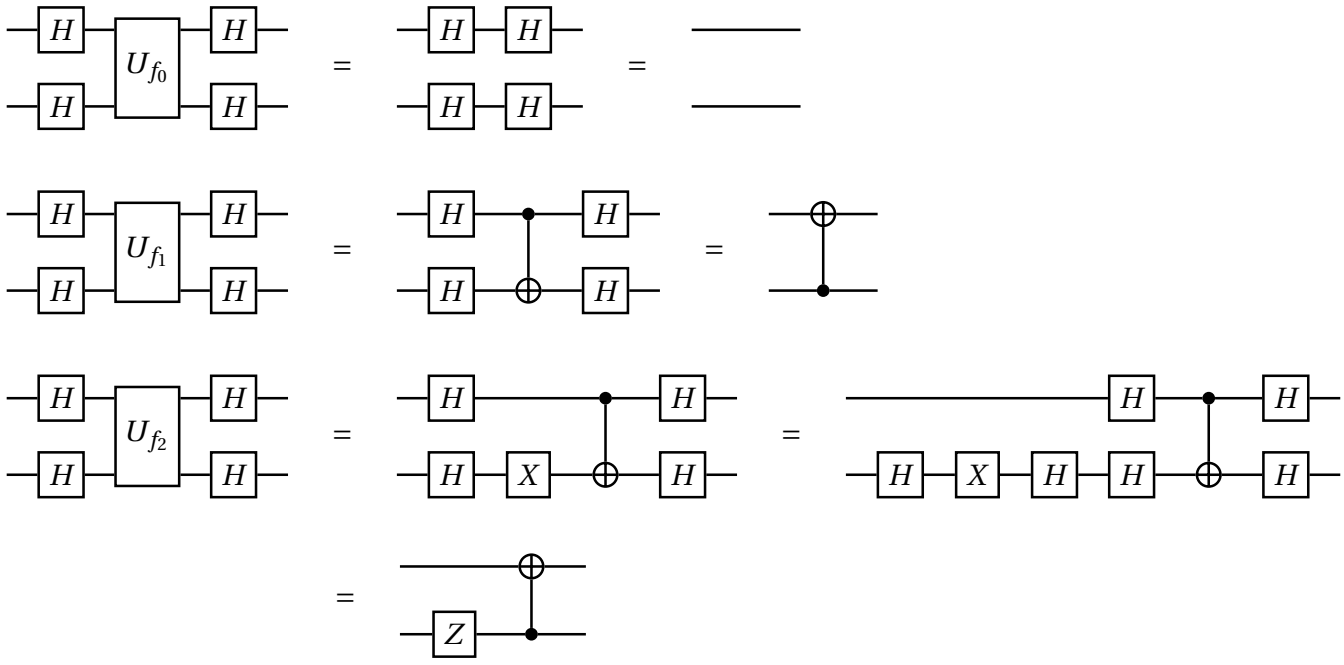
$$(H\otimes I)\,U_f\,(H\otimes H)\,(I\otimes X)\,|00\rangle = \begin{cases} \frac{1}{\sqrt{2}}|0\rangle\left(|f(0)\rangle - |1\oplus f(0)\rangle\right) & f(0)=f(1) \\ \frac{1}{\sqrt{2}}|1\rangle\left(|f(0)\rangle - |1\oplus f(0)\rangle\right) & f(0)\neq f(1) \end{cases} \tag{5.2.8}$$

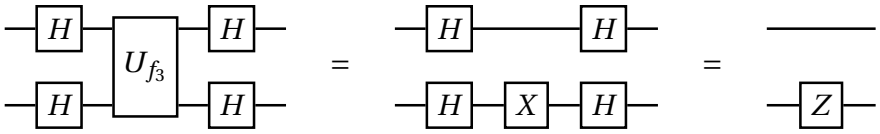
So then we just measure the state of the input qubit to determine whether $f(0) = f(1)$! The output qubit contains no useful information. The ability to learn whether $f(0) = f(1)$ comes at the cost of knowing anything about $f(0)$ or $f(1)$ individually. That's a little reminiscent of the Heisenberg uncertainty principle, for those of you who know about that.

We can use circuits and gate identities to see how the algorithm works. This will not provide much benefit here but will be useful for other problems. If we send the state $|1\rangle$ in as the output qubit, we can remove the initial NOT from the circuit that solves the Deutsch problem. And because at the end we only examine the top (input) bit, we might as well apply a Hadamard at the end of the bottom line to make the situation more symmetric. Those two steps make the circuit that implements the Deutsch algorithm look like this:



Now let' plug in the simple transformations from the table for U_f that appears after Eq. (5.2.3). The figure below (part or all of which may be on the next page) shows the results of using gate identities to simplify the circuits for the four different f_i.





It's easy to see that the input qubit is unaffected by either U_{f_0} or U_{f_3}, if the surrounding H's are included. But it's flipped by U_{f_1} and U_{f_2} if the bottom qubit is initially in the state $|1\rangle$ (as it is, because the of the X gate in Eq. (5.2.4)). Thus the constant functions give $|0\rangle$ for the final state of the input bit and the other two give $|1\rangle$, as we showed algebraically earlier.

5.3 Entanglement and Uncomputing

Our implementations of the simple U_f transformation involve only the two bits in the figures above. But a more complicated function will certainly entangle the input and output qubits with others needed to evaluate f. Entanglement will be necessary even in the Deutsch problem if f is doing something like seeing whether the millionth bits of $\sqrt{2}$ and $\sqrt{3}$ are the same. But entangling the input and output qubits with others can spoil the algorithm. Rieffel and Polak give a simple example of how that can happen. I present it here as well.

Suppose during the computation the input and output qubits (initially $|x\rangle$ and $|y\rangle$) get entangled with a third qubit initially in the state a). Let's assume, just to be specific, that U_f affects the three qubits works as follows:

$$U_f\,|x\rangle\,|a\rangle\,|y\rangle = |x\rangle\,|a\oplus x\rangle\,|y\oplus f(x)\rangle\,. \tag{5.3.1}$$

In other words, U_f does the same thing to the input and output qubits as before, but also alters another qubit in a way that depends on the input. What happens if we try to run the algorithm as before, by simply neglecting the extra bit? Well, let's trace the evolving state of the three-qubit system:

$$\begin{aligned}|000\rangle &\xrightarrow{(H\otimes I\otimes H)(I\otimes I\otimes X)} \frac{1}{2}\left(|000\rangle - |101\rangle + |100\rangle - |001\rangle\right)\\ &\xrightarrow{U_f} \frac{1}{2}\left(|0\rangle\,|0\rangle\,|f(0)\rangle - |1\rangle\,|1\rangle\,|1\oplus f(1)\rangle + |1\rangle\,|1\rangle\,|f(1)\rangle - |0\rangle\,|0\rangle\,|1\oplus f(0)\rangle\right).\end{aligned}$$

Now if $f(0) = f(1)$ this becomes

$$\frac{1}{2}\left(|0\rangle\,|0\rangle + |1\rangle\,|1\rangle\right)\left(|f(0)\rangle - |1\oplus f(0)\rangle\right), \tag{5.3.2}$$

while if $f(0) = 1\oplus f(1)$, it's now

$$\frac{1}{2}\left(|0\rangle\,|0\rangle - |1\rangle\,|1\rangle\right)\left(|f(0)\rangle - |1\oplus f(0)\rangle\right) \tag{5.3.3}$$

Now, though, when we apply H to the input bit we don't just get $|0\rangle$ or $|1\rangle$ like before because it's entangled with the other bit. Instead, if $f(0) = f(1)$ we get

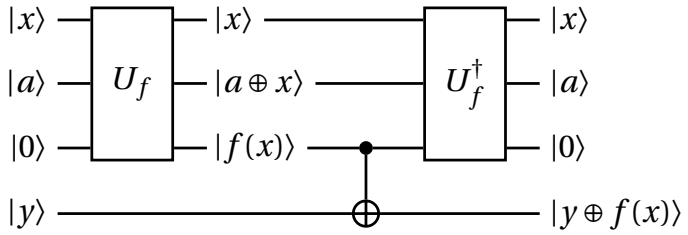
$$\frac{1}{2\sqrt{2}}\left(|0\rangle\,|0\rangle + |1\rangle\,|0\rangle + |0\rangle\,|1\rangle - |1\rangle\,|1\rangle\right)\left(|f(0)\rangle - |1\oplus f(0)\rangle\right), \tag{5.3.4}$$

while if $f(0) \neq f(1)$ we now get

$$\frac{1}{2\sqrt{2}}\left(|0\rangle|0\rangle + |1\rangle|0\rangle - |0\rangle|1\rangle + |1\rangle|1\rangle\right)\left(|f(0)\rangle - |1 \oplus f(0)\rangle\right), \tag{5.3.5}$$

In both cases, we have an equal chance of measuring the input bit to be in the state $|0\rangle$ and the state $|1\rangle$ and we can't distinguish constant from non-constant functions.

How can we get around this problem? We can make sure to "uncompute" the entangled bit. This figure shows how that works:



The extra qubit that gets entangled during the function evaluation is second from the top, just below the input. The trick to the circuit is the addition of another qubit (at the bottom) into which the result of the function is copied with the use of a controlled NOT. Then the function is reversed so that the other bits, including the extra one, return to their original values. Now the Hadamards that implement the Deutsch algorithm can be placed on the $|x\rangle$ and $|y\rangle$ lines, before and after everything else in this circuit, and the algorithm will work as originally advertised. It will also work if the incoming $|0\rangle$ state is $|1\rangle$ instead, which means that it will also work when that state is a superposition of $|0\rangle$ and $|1\rangle$.

Unfortunately, though, this need to uncompute when extra qubits are involved means that I lied to you earlier; the Deutsch algorithm actually offers no speedup if f is complicated enough to represent something like the billionth bit of $\sqrt{2+x}$. To compute that you'd need to entangle many auxiliary qubits with the input (if you didn't, the input wouldn't affect the computation) and you'd then have to uncompute with $U_f^\dagger$ after computing with U_f. So the algorithm would have to call something as complicated as U_f twice rather than just once, and you'd have no speedup.

5.4 Walsh-Hadamard

Suppose we have an n-qubit input register. The first move in a quantum algorithm, to obtain a superposition, is often to apply the "Walsh Hadamard" transformation $W \equiv H \otimes H \otimes \cdots \otimes H$ (n of these) to an n-qubit state. Here we derive a convenient expression for W applied to the ket containing any n-bit number x, which in binary can be written as $x_{n-1}x_{n-2}\ldots x_0$. We begin with the 1 bit case, $x = 0$ or 1:

$$\begin{aligned} H|x\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle + (-1)^x|1\rangle\right) \\ &= \frac{1}{\sqrt{2}}\sum_{y=0}^{1}(-1)^{xy}|y\rangle\,. \end{aligned} \tag{5.4.1}$$

So for an n- bit number,

$$\begin{aligned} W|x_{n-1}x_{n-2}\ldots x_0\rangle &= \frac{1}{\sqrt{2^n}}\left(|0\rangle+(-1)^{x_{n-1}}|1\rangle\right)\otimes\left(|0\rangle+(-1)^{x_{n-2}}|1\rangle\right)\otimes\cdots\otimes\left(|0\rangle+(-1)^{x_0}|1\rangle\right) \\ &= \frac{1}{\sqrt{2^n}}\sum_{y_{n-1}=0}^{1}(-1)^{x_{n-1}y_{n-1}}|y_{n-1}\rangle\sum_{y_{n-2}=0}^{1}(-1)^{x_{n-2}y_{n-2}}|y_{n-2}\rangle\cdots\sum_{y_0=0}^{1}(-1)^{x_0y_0}|y_0\rangle \\ &= \frac{1}{\sqrt{2^n}}\sum_{y_{n-1},y_{n-2}\ldots y_0=0}^{1}(-1)^{x_{n-1}y_{n-1}+x_{n-2}y_{n-2}+\cdots+x_0y_0}|y_{n-1}y_{n-2}\cdots y_0\rangle \\ &= \frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1}(-1)^{x\cdot y}|y\rangle\,, \end{aligned} \tag{5.4.2}$$

where in the last line y is an n-bit number and $x\cdot y$ is what you'd get you thought of the n-bit binary numbers as n-component vectors:

$$x\cdot y \equiv x_0y_0+x_1y_1+\cdots+x_{n-1}y_{n-1}\,. \tag{5.4.3}$$

5.5 Bernstein-Vazirani Problem

Suppose that an oracle produces an unknown n-bit number u and that you can evaluate the function $f(x)=u\cdot x \pmod 2$ for any n-bit x. Your goal is to identify u. With a regular computer you would get the m^{th} bit of u by computing $u\cdot x \pmod 2$ with x the number that that has a 1 in the m^{th} bit and 0's everywhere else. You'd have to call $f(x)$ n times to determine u. With a quantum computer you'd need to call U_f, the quantum version of f, only once. What??

The manipulations are very similar to those in the Deutsch-Jozsa algorithm, which was or will be on your homework. I'll go through them here. First note what happens when you apply U_f to a computational-basis input after flipping the output qubit and applying H to it, as in the Deutsch algorithm:

$$\begin{aligned} U_f(I\otimes H)|x\rangle_n|1\rangle &= U_f|x\rangle_n|-\rangle \\ &= U_f|x\rangle_n\frac{1}{\sqrt{2}}(|0\rangle-|1\rangle) \\ &= \frac{1}{\sqrt{2}}|x\rangle_n\left(|f(x)\rangle-|1\oplus f(x)\rangle\right) \\ &= \begin{cases} \frac{1}{\sqrt{2}}|x\rangle_n(|0\rangle-|1\rangle) & f(x)=0 \\ \frac{1}{\sqrt{2}}|x\rangle_n(|1\rangle-|0\rangle) & f(x)=1 \end{cases} \\ &= (-1)^{f(x)}|x\rangle_n|-\rangle\,. \end{aligned} \tag{5.5.1}$$

I've put a subscript n on the input ket to emphasize that it's an n-bit state. If we do the usual trick of using the Walsh-Hadamard transformation to put the input into a super-

position of all possible numbers before applying U_f, we get

$$\begin{aligned} U_f\,(W\otimes H)\,|0\rangle_n\,|1\rangle &= \frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1} U_f\,|y\rangle_n\,|-\rangle \\ &= \frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1}(-1)^{f(y)}\,|y\rangle_n\,|-\rangle\,. \end{aligned} \tag{5.5.2}$$

Now we've got a sum of states corresponding to different input numbers y, with signs that depend on $f(y)$. We need sum cancellations in the sum over states $|y\rangle$ for u to be singled out, but there aren't any; all terms are present. We thus need to play around a little more. Suppose we apply the Walsh-Hadamard transformation to the input register *after* the action of U_f, as well as before. Then, using (5.4.2), we obtain

$$\begin{aligned} (W\otimes I)\,U_f\,(W\otimes H)\,|0\rangle_n\,|1\rangle &= (W\otimes I)\,\frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1}(-1)^{f(y)}\,|y\rangle_n\,|-\rangle \\ &= \frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1}(-1)^{f(y)}\left(W\,|y\rangle_n\right)|-\rangle \\ &= \frac{1}{2^n}\sum_{x,y=0}^{2^n-1}(-1)^{x\cdot y+f(y)}\,|x\rangle_n\,|-\rangle\,. \end{aligned} \tag{5.5.3}$$

Now the idea, from one point of view anyway, is that all the terms in the sum over y above will cancel (i.e. interfere destructively) unless $x=u$. Let us see, recalling the $f(y)=u\cdot y \pmod 2$, how that works:

$$\begin{aligned} \sum_{y=0}^{2^n-1}(-1)^{x\cdot y+f(y)} &= \sum_{y=0}^{2^n-1}(-1)^{y\cdot x+y\cdot u \pmod 2} \\ &= \sum_{y=0}^{2^n-1}(-1)^{y\cdot(x\oplus u)} \\ &= \underbrace{\sum_{y_0=0}^{1}(-1)^{y_0(x_0\oplus u_0)}}_{\substack{2 \text{ if } x_0=u_0\\ 0 \text{ otherwise.}}}\sum_{y_1=0}^{1}(-1)^{y_1(x_1\oplus u_1)}\ldots \\ &= \begin{cases} 2^n & x_i=u_i \text{ for all } i \\ 0 & \text{otherwise} \end{cases}. \end{aligned} \tag{5.5.4}$$

(The journey from the end of the first line to the beginning of the second is mentioned in Rieffel and Polak's box 7.1, and you'll prove it in the next homework.) The whole expression vanishes if *any* of the bits in x is different from the corresponding bit in u!

Now, when we plug all this into (5.5.3) we have

$$\begin{aligned}(W\otimes I)\,U_f\,(W\otimes H)\,|0\rangle_n\,|1\rangle &= \frac{1}{2^n}\sum_{x,y=0}^{2^n-1}(-1)^{x\cdot y+f(y)}\,|x\rangle_n\,|-\rangle \\ &= \frac{1}{2^n}2^n\,|u\rangle_n\,|-\rangle \\ &= |u\rangle\,|-\rangle\,.\end{aligned} \tag{5.5.5}$$

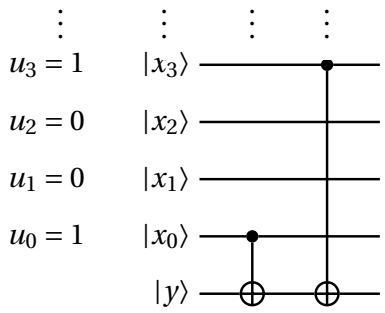
Finally, just to make things symmetric, we an add an H to the output bit at the end, yielding

$$(W\otimes H)\,U_f\,(W\otimes H)\,|0\rangle_n\,|1\rangle = |u\rangle\,|1\rangle\,. \tag{5.5.6}$$

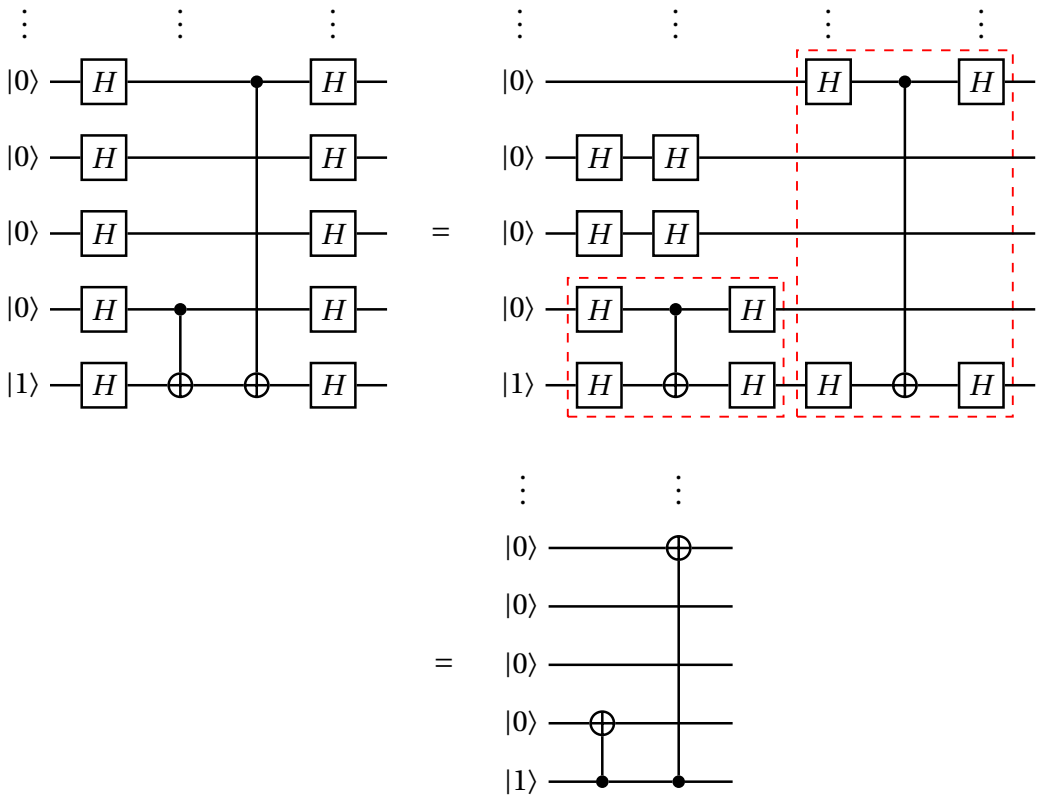
So measuring the input state yields the number u. Bizarre, man.

Eq. (5.5.5) makes it seem as if the algorithm works by making all the terms but one in a superposition cancel each other out. In the many-worlds interpretation, the worlds all do their calculations, then combine them in such a way so as to leave just one result. But there's a way of looking at the algorithm that makes it seem like something else is responsible.

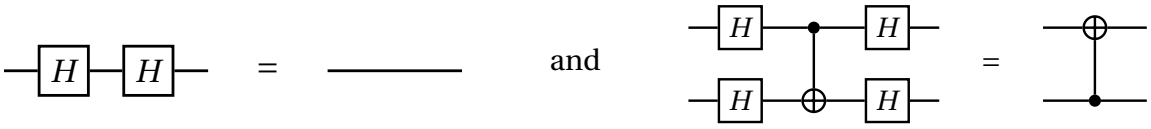
Consider what could be a four-qubit number $u = 1001$ or, more generally, the last four qubits of an n-qubit number $u = u_{n-1}u_{n-2}\ldots1001$. The circuit that implements U_f for $f(x) \equiv u\cdot x \mod 2$ is shown below:



It adds 1 modulo 2 every time it encounters a bit of u that is 1 rather than zero by having each 1-bit control a not on the output. The Bernstein-Vazirani algorithm surrounds U_f with Hadamards on all the input qubits and the output qubit, as the first figure below shows.



If we slide the H gates along their wires (legally), we can group them as shown in the second figure and then use two of the identities that you proved earlier:



to get the last line in the figure.

Now what does this simplified version of the circuit do? It just puts a 1 in every slot where u has a 1, thus yielding $|u\rangle$ in the input register, just like we found before. This circuit explanation of the algorithm, however, simply relies on the fact that surrounding a controlled NOT with Hadamards reverses the control and target bits. It doesn't involve interference of 2^n terms with canceling phases, etc. The moral of the story, according to Rieffel and Polak, is that "how" quantum computing achieves the results it does is not always an easy matter to specify uniquely. The same is true of how quantum mechanics works in general.

5.6 Simon's Problem

Simon's problem is this: Suppose you have an oracle that evaluates a 2-to-1 function $f(x)$ on n-bit numbers (so that $f(x)$ itself is an $n-1$-bit number). In other words, there are always two values of x that yield the same $f(x)$. Suppose, further, that f has the property: $f(x) = f(x \oplus a)$ for all x and for some particular unknown value of a. (Here

$\oplus$ is the "bitwise addition" operator; you apply it to each bit in the sum of two multi-bit numbers.) Find a with as few calls to U_f as possible.

Here's a simple example. Consider the 2-to-1 function below:

x_1	x_0	$f(x)$
0	0	0
0	1	1
1	0	0
1	1	1

Here $a = 2$, which is the string 10 in binary.

Simon's algorithm, though still not particularly useful, is more like Shor's algorithm than the others we've looked at. For one thing, it's a kind of period-finding algorithm for bit-wise addition: the condition $f(x) = f(x \oplus a)$ says that f has period a if we agree to add numbers bit-wise in defining "period" (note that $x \oplus a \oplus a = x$). Shor's algorithm finds periods for the usual arithmetic, which allows to factoring the product of two large numbers, as we'll see. Also, both Simon's and Shor's algorithms find their periods quickly but only with very high probability — you can make the probability very close to one by repeating the algorithm a small number of times. Finally, both algorithms are exponentially faster than their classical counterparts.

How would you solve Simon's problem classically. You'd need to compute $f(x^{(1)}), f(x^{(2)}), \ldots$, for random n-bit numbers $x^{(i)}$ until you found two of them, say $x^{(i)}$ and $x^{(j)}$, with $f(x^{(i)}) = f(x^{(j)})$. Then you'd find a from $x^{(j)} = x^{(i)} \oplus a$, which is equivalent to $a = x^{(i)} \oplus x^{(j)}$. After m x's, assuming you hadn't yet found $x^{(i)}$ and $x^{(j)}$, you'd have eliminated $\frac{1}{2}m(m-1)$ (the number of independent pairs you can form from m objects) values for a. The chances of finding a are thus small unless $\frac{1}{2}m(m-1) \approx 2^n$, i.e. $m \approx 2^{\frac{n}{2}}$.

With Simon's algorithm, you only need about $m \approx n$ calls to U_f to determine a to high probability (say 0.999999). Here's how the algorithm works:

First, apply W to the input and note/recall that

$$|\Psi\rangle \equiv U_f (W \otimes I)\,|0\rangle_n\,|0\rangle_{n-1} = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle_n\,|f(x)\rangle_{n-1}\,. \tag{5.6.1}$$

Now measure the output bits. When you do, $|\Psi\rangle$ collapses to

$$|\Phi\rangle = \frac{1}{\sqrt{2}}\left(|\overline{x}\rangle + |\overline{x} \oplus a\rangle\right)|f(\overline{x})\rangle\,, \tag{5.6.2}$$

(where I'm now dropping the subscripts n and $n-1$) for some completely random $\overline{x}$. This looks really promising, because if you knew both $\overline{x}$ and $\overline{x} \oplus a$, you could add them to determine a. But without making many copies of this particular state (which the no cloning theorem says we cannot do) you can't determine both $\overline{x}$ and $\overline{x} \oplus a$. We need to be smarter to extract a.

You can be smarter, as in Deutsch's algorithm, by forgetting about trying to extract either $\overline{x}$ or $\overline{x} \oplus a$ itself. Instead, try to shift the dependence on $\overline{x}$ to an irrelevant phase.

You can do this by applying W to the input register, so that from (5.4.2) you then get

$$\begin{aligned}(W\otimes I)\,|\Phi\rangle &= \frac{1}{\sqrt{2^{n+1}}}\sum_{y=0}^{2^n-1}\left[(-1)^{\overline{x}\cdot y}+(-1)^{(\overline{x}+a)\cdot y}\right]|y\rangle\,|f(\overline{x})\rangle \\ &= \frac{1}{\sqrt{2^{n+1}}}\sum_{y}(-1)^{\overline{x}\cdot y}\underbrace{\left(1+(-1)^{a\cdot y}\right)}_{0\text{ or }2}|y\rangle\,|f(\overline{x})\rangle \\ &= \frac{1}{\sqrt{2^{n-1}}}\sum_{y,\,a\cdot y\text{ even}}(-1)^{\overline{x}\cdot y}\,|y\rangle\,|f(\overline{x})\rangle\,.\end{aligned} \tag{5.6.3}$$

The sum goes only over those values of y that are "orthogonal" (mod 2) to a:

$$a_0y_0+a_1y_1+\cdots+a_{n-1}y_{n-1}=0 \mod 2\,. \tag{5.6.4}$$

You measure the input to determine y. Repeat as many times as you need to, and that's it for the quantum computer.

What does it mean to repeat as many times as you need to? Well, the first time you carry out the steps above, unless you collapse to $y=0$, you can determine one of the a_i in terms of the other one. Suppose you have a four-bit a and you collapse to $|y=1011\rangle$. Then you know that

$$a_0+a_1+a_3=0 \mod 2\,. \tag{5.6.5}$$

Since all the a_i are either 0 or 1, you can conclude, e.g., that

$$a_0=a_1+a_3 \mod 2\,, \tag{5.6.6}$$

reducing the number of unknown bits from 4 to 3. When you run the quantum circuit again, unless you're unlucky enough to collapse into $|0000\rangle$ or $|1011\rangle$ (the probability of which is $1/4$), you will get another independent constraint on the bits of a, reducing the number of unknown bits to two. Two more runs, and if you have good luck you'll have a. If you do have some bad luck along the way you just run another few times. For an n-bit number a, it's possible to show that with $n+x$ runs, the probability of acquiring enough information to determine a is

$$\text{Prob.(success)} > 1-\frac{1}{2^{x+1}}\,. \tag{5.6.7}$$

The odds are greater than a million to one that you'd have enough info with $n+20$ runs, no matter how large n may be. This is a lot better than the $2^{n/2}$ evaluations of $f(x)$ you'd need with a regular computer.

The analysis that leads to (5.6.7) is complicated. In that it is like Shor's algorithm, which we take up next.

6 Shor's Algorithm

6.1 Modular Arithmetic

Shor's algorithm provides a way to use a quantum computer to factor numbers. Factoring is closely related to modular arithmetic and period finding. I explain the connection here, starting with modular arithmetic.

Given any positive integer x and another number N, we can always write

$$x = kN + y, \tag{6.1.1}$$

with $k \geq 0$ and $y < N$, and there is only one way do so, i.e. k and y are unique. In modular arithmetic the "remainder" y is equivalent to x:

$$y = x \mod N. \tag{6.1.2}$$

Here's an example of an equality:

$$2 = 5 = 8 = 11 \mod 3. \tag{6.1.3}$$

The reason:

$$\begin{aligned} 2 &= 0 \cdot 3 + 2 \\ 5 &= 1 \cdot 3 + 2 \\ 8 &= 2 \cdot 3 + 2 \\ 11 &= 3 \cdot 3 + 2. \end{aligned} \tag{6.1.4}$$

All the remainders are 2.

Now let's look what happens when we repeatedly exponentiate a number modularly. Consider $3^n \mod 5$ for $n = 1, 2, 3, 4$.

$$\begin{aligned} 3^1 &= 3 \mod 5 \\ 3^2 &= 9 = 4 \mod 5 \\ 3^3 &= 27 = 2 \mod 5 \\ 3^4 &= 81 = 1 \mod 5, \end{aligned} \tag{6.1.5}$$

and after that the sequence repeats. The next term, for example, is $3^5 = 243 = 3 \mod 5$. Is it obvious that the pattern repeats? Well, if it isn't you can note that if you're multiplying 3 by a number larger than 5, you can drop everything but the remainder when multiplying, i.e. you note that

$$\begin{aligned} 3^5 &= 3 \times 3^4 \\ &= 3 \times 81 \\ &= 3 \times (18 \times 5 + 1) \\ &= 54 \times 5 + 3 \times 1 \\ &= 3 \mod 5 \\ &= 3 \times 1 \mod 5. \end{aligned} \tag{6.1.6}$$

You can drop the first part because it contains a multiple of 5 before the multiplication by 3, and so still contains a multiple of 5 after the multiplication by 3. It therefore contributes nothing to the remainder, which is all that matters for the "mod" operation. So indeed, the series repeats, and $3^n \mod 5$ is a sequence with period $r = 4$.

You could pick another number to exponentiate, e.g., 4:

$$\begin{aligned} 4^1 &= 4 \mod 5 \\ 4^2 &= 16 = 1 \mod 5, \end{aligned}$$

and the sequence repeats with period $r = 2$. Does any number that you pick lead to a repeating sequence when exponentiated? Well, there are only 5 distinct numbers that are less than 5 (0, 1, 2, 3, and 4), so that a list of the first 6 powers of any number $a \mod 5$ that is less than 5 must contain at least one of the numbers twice. That means that $a^y = a^x \mod 5$ for some x and y with $y > x$. That in turn means that $a^y - a^x = a^x(a^{y-x} - 1)$ is a multiple of 5. As long as a is not zero, a^x is not a multiple of 5 (because it's a multiple only of a or factors of a.) So $a^{y-x} - 1$ must be a multiple of 5, which means that

$$a^{y-x} = 1 \mod 5. \tag{6.1.7}$$

If some nonzero power of a is equal to one modulo 5, then there must be a smallest power of a that is equal to one modulo 5. If that power is r, then $a^n \mod 5$ is periodic with period r. So yes, $a^n \mod 5$ is periodic for any a. And that is true whether you're doing modulo-5 arithmetic or modulo-N arithmetic, where N is any other prime number. What if N is not prime? Well, as long as a and N share no common factors, the argument still works. What happens if they do share common factors? Then a^x could (sometimes) be a multiple of N and the argument doesn't go through.

6.2 Factoring

6.2.1 Factoring is Period Finding

So what does this have to do with factoring? Well, suppose you are trying to factor a large number N that is a product of two large primes p and q. You choose a number a less than N that has no common factors with it — that will be overwhelmingly likely because N has only two factors. If you're able to compute the period r of the sequence $a^n \mod N$ (which is what the quantum part of Shor's algorithm does) then you know that a^r is a multiple of N, plus 1. That means that $a^r - 1$ is a multiple of N.

Suppose, further, that r happens to be even (if it isn't we compute the period for another value of a). Then $a^{r/2}$ is an integer and $\left(a^{r/2}+1\right)\left(a^{r/2}-1\right)$ is a multiple of N (because it's the same as $a^r - 1$). Now suppose, still further, that neither of these factors is itself a multiple of N. The second factor $a^{r/2}-1$ could never be a multiple of N because that would require $a^{r/2} = 1 \mod N$, and we know that $x = r$ is the smallest number such that $a^x = 1 \mod N$. But $a^{r/2} + 1$ could be a multiple of N and for our scheme to work we have to be lucky enough so that it isn't. If we are that lucky, then neither $a^{r/2} - 1$ nor $a^{r/2}+1$ is a multiple of N, but their product is! That means that $a^{r/2}-1$ shares one factor (say p) with N and $a^{r/2} + 1$ shares the other. And in fact, because the only factors of N are p and q, p is the greatest common divisor (g.c.d.) of $a^{r/2}-1$ and N, and q is the g.c.d. of $a^{r/2} + 1$ and N.

Then we're left with the task of finding the g.c.d.'s. That's easy, as we'll see shortly. But first, let's see how this method works for the problem of factoring 15. I'll choose a random a that has no factors in common with 15, say $a = 7$. Now, from the equivalence relations (the mod 15 at the end of each relation means that all the equals signs refer to

modulo 15 equivalences)

$$\begin{aligned} 7^1 &= 7 \mod 15 \\ 7^2 &= 49 = 4 \mod 15 \\ 7^3 &= 7 \times 4 = 28 = 13 \mod 15 \\ 7^4 &= 7 \times 13 = 91 = 1 \mod 15, \end{aligned} \tag{6.2.1}$$

we see that $r = 4$. And we got lucky; 4 is even. So we factor $7^4 - 1$, which as we can see from the above is a multiple of 15, into

$$7^4 - 1 = \left(7^2 - 1\right)\left(7^2 + 1\right). \tag{6.2.2}$$

We need to check whether $7^2 + 1 = 50$ is a multiple of 15; it isn't. So we know that $7^2 - 1 = 48$ is a multiple of one of the factors of 15 and $7^2 + 1 = 50$ is a multiple of the other. Now 3 is the g.c.d. of 48 and 15, and 5 is the g.c.d. of 50 and 15. So we know that the prime factors of 15 are 3 and 5. Yay!

What about for much larger numbers N, say those on the order of 10^{400}? There are still a few loose ends to clean up in our procedure before tackling those. The first is that we have to get lucky in two ways for our procedure to work: r must be even, and $a^{r/2} + 1$ must not be a multiple of N. Though I won't show it (see, e.g., the appendix of Mermin), the chances that we have both pieces of luck are at least 50%. The other loose end is the problem of finding the g.c.d.'s of $a^{r/2} - 1$ and N, and of $a^{r/2} + 1$ and N. I said above that that's easy, but I have to show you why.

6.2.2 Euclid's Algorithm

The problem of finding the g.c.d. of two numbers, once you know they have factors in common, was solved by Euclid, and the method is known as Euclid's algorithm. Here it is, combined with an explanation of why it works:

Suppose you want to find the g.c.d. of f and c, with $f > c$. Start by defining new numbers

$$\begin{aligned} f' &= c \\ c' &= f - [f/c]c, \end{aligned} \tag{6.2.3}$$

where $[f/c]$ means the largest integer that is less than f/c, i.e. the part that is not the remainder when you divide f by c. Note that every common factor of f and c is also a common factor of f' (because $f' = c$) and c' (because it's a factor of both f and c, and so a factor of f minus an integer of c). The reverse is also true: every common factor of f' and c' is a factor both of f, because

$$\begin{aligned} f &= c' + [f/c]c \\ &= c' + [f/c]f', \end{aligned} \tag{6.2.4}$$

and of c, because $c = f'$. Also $f' < f$ because $c < f$, and $c' < c$. Why this last inequality? Well, let

$$f = nc + x, \quad x < c, \tag{6.2.5}$$

for some integer n. Then

$$\frac{f}{c} = n + \frac{x}{c}$$

so that

$$[f/c] = n$$

and

$$\begin{aligned} c' &\equiv f - [f/c]c \\ &= nc + x - nc \\ &= x < c. \end{aligned} \tag{6.2.6}$$

So, indeed, $f' < f$ and $c' < c$, and the common factors of f' and c' are identical with those of f and c.

Now you redefine f and c to be f' and c' and repeat the procedure. You keep doing this until $c' = 0$, which must happen eventually (if not you could do it again). Call the last f and c before then f_0 and c_0. These two numbers have the same common factors as f and c, and f_0 is a multiple of c_0 (because of (6.2.3) and the fact that the next c' is zero). Since c_0 is the greatest factor of itself, it is the g.c.d. of f_0 and c_0, and thus the g.c.d. of f and c. And that's that.

6.3 RSA Encryption

In the RSA scheme, Bob gives Alice a large number N that is the product of two primes p and q of his choice (that nobody else knows) and an "encryption number" (or "coding number") c. Alice has a message in the form of a number a, encrypts it by computing $b \equiv a^c \mod N$, and sends b to Bob. Bob has taken advantage of some number theory to compute for himself a special number d with the property that $b^d = a \mod N$, no matter what a is. He raises Alice's encoded message b to the d^{th} power and extracts a. Nobody else can figure out what d is without knowing p and q and so nobody else can extract a.

We need a little more number theory to understand why d exists and how to get it, in particular, what's called Fermat's "little theorem."

6.3.1 Fermat's Little Theorem and Extension

The theorem: For any prime number p and $a < p$ (or more generally, for a not divisible by p),

$$a^{p-1} = 1 \mod p. \tag{6.3.1}$$

To prove this, we first will have to show that the set of remainders mod p of the numbers $a, 2a, 3a, \ldots (p-1)a$, all mod p, are exactly the set $1, 2, \ldots p-1$, though in a different order

in general. Consider, for example, $a = 3$ and $p = 7$. We get

$$\begin{aligned} 1 \times 3 &= 3 \\ 2 \times 3 &= 6 \mod 7 \\ 3 \times 3 &= 2 \mod 7 \\ 4 \times 3 &= 5 \mod 7 \\ 5 \times 3 &= 1 \mod 7 \\ 6 \times 3 &= 4 \mod 7 \end{aligned} \tag{6.3.2}$$

so all the numbers less than p are indeed produced by the first six multiples of a.

Why does this hold for arbitrary p and a? Well, suppose that it didn't. Then there would be two different numbers x and y less than p such that

$$xa = ya \mod p \quad \Longrightarrow \quad (x-y)a = 0 \mod p\,, \tag{6.3.3}$$

which would mean that $(x-y)a$ was divisible by p. Since p is prime, and factorization into primes is unique, that in turn would mean that either $(x-y)$ or a individually (or both) would contain p as a factor. We're assuming, though, that $a < p$ (or, more generally, that a is not divisible by p) so it can't have p as a factor. Also, x and y are less than p, so $|x-y| < p$, and $x-y$ also can't have p as a factor. So our assumption that $x \neq y$ cannot be right, i.e. it must be the case that if $xa = ya \mod p$, then $x = y \mod p$. Thus all numbers $a, 2a, \ldots (p-1)a$ must be different from one another, so that the set of them is just the set of numbers between 1 and $p-1$ in a different order.

This last fact implies that

$$\begin{aligned} a \times 2a \times 3a \cdots \times (p-1)a &= 1 \times 2 \times 3 \cdots \times (p-1) \mod p \\ \Longrightarrow \quad a^{p-1}(p-1)! &= (p-1)! \mod p\,. \end{aligned} \tag{6.3.4}$$

If we could "divide" both sides by $(p-1)!$ we'd have proved what we want. Can we? Modular division can be tricky, so we have to look a little closer. Equation (6.3.4) implies that $\left(a^{p-1}-1\right)(p-1)!$ is a multiple of p, so that either $a^{p-1}-1$ or $(p-1)!$ has p as a factor. But $(p-1)!$ certainly does not have p as a factor, so $a^{p-1}-1$ must. That means that

$$a^{p-1} - 1 = 0 \mod p\,, \tag{6.3.5}$$

which implies that, indeed, in this case modular division works as expected and we could have removed the $(p-1)!$ from Eq. (6.3.4) to obtain

$$a^{p-1} = 1 \mod p\,. \quad \checkmark \tag{6.3.6}$$

That's the little theorem.

RSA encryption makes use of the following generalization of this theorem. If a is divisible by neither of *two* primes p and q, then no power of a is also divisible by either prime, because the factors of a^n are the same as those of a. In particular, a^{p-1} is not divisible by q, and Fermat's little theorem then implies that

$$\left(a^{p-1}\right)^{q-1} = a^{(q-1)(p-1)} = 1 \mod q\,, \tag{6.3.7}$$

and, similarly, that

$$\left(a^{q-1}\right)^{p-1} = a^{(q-1)(p-1)} = 1 \mod p. \tag{6.3.8}$$

So $a^{(q-1)(p-1)} - 1$ is a multiple of *both* p and q. It is therefore a multiple of pq as well (when you break it up into prime factors, p and q appear multiplied by one another), and so

$$\boxed{a^{(q-1)(p-1)} = 1 \mod pq}. \tag{6.3.9}$$

A quick example: Let $a = 5$, $p = 2$ and $q = 3$. Then the (6.3.9) says that

$$5^{2\times 1} = 1 \mod 6, \tag{6.3.10}$$

which indeed is correct.

Note, finally, that if $x = 1 \mod p$, then, as a consequence of what you will show about modular multiplication in your homework, $x^s = 1 \mod p$ for any integer s. So (6.3.9) implies that

$$a^{s(q-1)(p-1)} = 1 \mod pq, \tag{6.3.11}$$

for any integer s, and

$$a^{1+s(q-1)(p-1)} = a \times a^{s(q-1)(p-1)} \mod pq. \tag{6.3.12}$$

With (6.3.11), that means that

$$\boxed{a^{1+s(q-1)(p-1)} = a \mod pq}. \tag{6.3.13}$$

This relation is the basis of RSA encryption.

Now we just need to say a few things about modular inverses and we'll be ready to lay out the RSA protocol.

6.3.2 Modular Inverse

The inverse mod N of a number a is defined to be another number b with the property $ab = 1 \mod N$. Does such an inverse always exist? Our proof above that the sequence $a^n \mod N$ is periodic if a and N share no factors implies that in that case, at least, a has a modular inverse. The reason is that if a^n has period r, then

$$a^r = aa^{r-1} = 1 \mod N. \tag{6.3.14}$$

That means that $b = a^{r-1}$ is the inverse modulo N of a.

Now let the number d be the inverse of another number c modulo $(p-1)(q-1)$, where p and q are prime as above and c shares no common factors with $(p-1)(q-1)$:

$$cd = 1 \mod (p-1)(q-1). \tag{6.3.15}$$

This relation means that for some number s,

$$cd = 1 + s(p-1)(q-1). \tag{6.3.16}$$

Eq. (6.3.13) then implies that

$$a^{cd} = a \mod pq. \tag{6.3.17}$$

So if

$$b \equiv a^c \mod pq,$$

then

$$\boxed{b^d = a \mod pq}. \tag{6.3.18}$$

6.4 RSA procedure

In this section I'm usually using "x mod M" (where M is any integer) to mean the remainder when x is divided by M. The number's we'll be working with will all be less than (or equal to) $N \equiv pq$. Here's how Alice and Bob use all we've just discussed to encode and decode a message:

1. Bob chooses two perhaps 1000-digit primes p and q and then supplies Alice with the number $N \equiv pq$ and a public "key" c that has no common factors with $(p-1)(q-1)$.

2. Bob then uses a straightforward extension of Euclid's algorithm that we won't go over to find $d \equiv c^{-1} \mod (p-1)(q-1)$. (Note: here c^{-1} is the modular inverse of c, not 1 divided by c.)

3. Through some simple code that everyone might know, Alice's message can be turned into a number less than N we call a. (If the message is too long for that, it can be broken up into pieces). Alice uses her knowledge of c to calculate $b \equiv a^c \mod N$ (the equals sign with three lines here means that b is the "remainder," i.e. that $b < N$), and sends b to Bob.

4. Bob then computes the remainder of $b^d \mod N$, which from (6.3.18) is Alice's original string a. Yay!

5. Even if Eve knows c, she can't compute d because she doesn't know p or q so that she can't know the value of $(p-1)(q-1)$. If she could factor N she could find d the same way Bob does, and then compute Alice's string a if she intercepted b, again the same way Bob does.

And that's RSA encryption!

Here's a simple example with small numbers. Let's choose $p = 3$ and $q = 7$. With these choices, $N \equiv pq = 21$ and $(p-1)(q-1) = 2 \times 6 = 12$. Let's choose the public key c to be 5, which has no common factors with 12. Then d is the inverse of 5 modulo 12, i.e $d = 5$ (because $5 \times 5 = 25 = 2 \times 12 + 1$). Finally, suppose Alice's message is $a = 4$.

OK, Alice computes $b = 4^5 \mod 21$:

$$\begin{aligned} b &= 4^5 \mod 21 \\ &= 1024 \mod 21 \\ &= 48 \times 21 + 16 \mod 21 \\ &= 16\,. \end{aligned} \tag{6.4.1}$$

Bob uses his knowledge of $d = 5$ and computes $b^d \mod 21$:

$$\begin{aligned} b^d \mod 21 &= 16^5 \mod 21 \\ &= 1048576 \mod 21 \\ &= 49932 \times 21 + 4 \mod 21 \\ &= 4\,. \end{aligned} \tag{6.4.2}$$

So he recovers Alice's message $a = 4$.

6.5 Discrete Fourier Transforms

The Fourier transform is a way of decomposing functions into sets of complex exponentials (which are made up of sines and cosines because of Euler's identity) with a different periods. We'll be taking the Fourier transform of functions ($a^n \mod N$) that are already periodic. The Fourier transform of a periodic function is particularly simple, as we'll see.

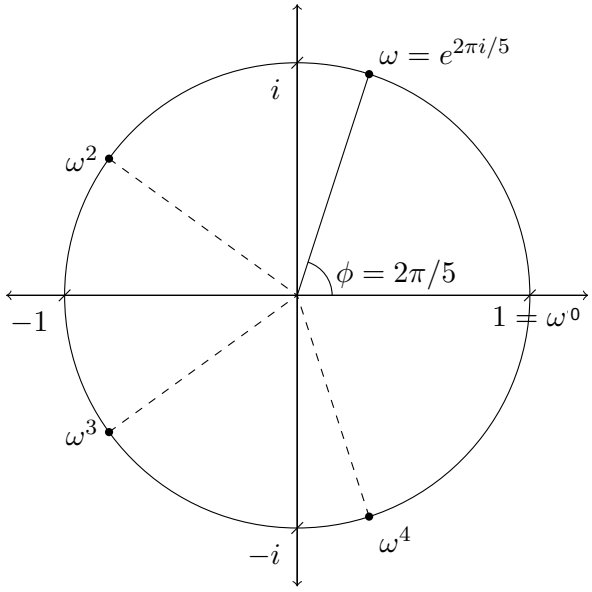
Because we're working with discrete numbers, we have to use a discrete version of the Fourier transform. Our function f(y) is a function of integers y less than some $N = 2^n$, and can thus be represented by a finite set of numbers c_y, where $y = 0, 1, \ldots N = 2^n$. The discrete Fourier transform of the "function" c_y is another discrete function $\tilde{c}_x$, defined as

$$\tilde{c}_x = \sum_{y=0}^{N-1} c_y \, e^{2\pi i x y / N}\,. \tag{6.5.1}$$

Before starting to analyze this, let's look at the complex exponentials that multiply c_y. They are powers of the number $e^{2\pi i/N}$, which is on the unit circle in the complex plane, making an angle $2\pi/N$ with the x axis (In this discussion of complex exponentials, I'm not requiring that N be some power of 2.) If you square it you get

$$\left(e^{2\pi i/N}\right)^2 = e^{4\pi i/N}\,, \tag{6.5.2}$$

a number represented by a point making an angle $4\pi/N$ with the x axis, so that it lies twice as far around the circle as the number that was squared. If you keep raising the original number to successively high powers, you get a set of points that looks like this (for $N = 5$):



What happens if you add all these numbers. It looks like they'll sort of cancel out. In fact they exactly do. We're about to show that

$$\sum_{y=0}^{N-1} e^{2\pi i x y/N} = \begin{cases} N & x = kN \text{ (integer } k\text{)} \\ 0 & \text{otherwise} \end{cases} . \tag{6.5.3}$$

The cancellation of the complex numbers in the figure above, when added, is a special case of this relation, with $x = 1$ and $N = 5$.

Why is (6.5.3) correct? Consider first the case $x = kN$. In that case

$$\begin{aligned} \sum_{y=0}^{N-1} e^{2\pi i x y/N} &= \sum_{y=0}^{N-1} e^{2\pi i y k} \\ &= \sum_{y=0}^{N-1} \left(e^{2\pi i k}\right)^y \\ &= \sum_{y=0}^{N-1} 1^y \\ &= N. \quad \checkmark \end{aligned} \tag{6.5.4}$$

The idea here is that all the terms in the sum are 1 and so add constructively. If $x \neq kN$, however, then they cancel as in the figure. Using the relation

$$\sum_{y=0}^{N-1} a^y = \frac{1-a^N}{1-a}, \tag{6.5.5}$$

we have

$$\begin{aligned}\sum_{y=0}^{N-1} e^{2\pi i x y/N} &= \frac{1-e^{2\pi i x N/N}}{1-e^{2\pi i x/N}} \\ &= \frac{1-e^{2\pi i x}}{1-e^{2\pi i x/N}} \\ &= \frac{1-1}{1-e^{2\pi i x/N}} \\ &= 0\,, \quad \checkmark\end{aligned} \tag{6.5.6}$$

where the last equality follows from the fact that x/N is not an integer so that $e^{2\pi i x/N} \neq 1$.

This "interference," constructive or destructive depending on the value of x, is the reason that Fourier transforms are useful for diagnosing periodicity. Let's see how that works. Consider the Fourier transform of a function c_y that has a period that exactly divides N, so that the function has an integer number of periods between 0 and $N-1$. Thus the function has period $r = N/m$ for some integer m. Then $c_y = c_{y+r}$ and the Fourier transform is

$$\begin{aligned}\tilde{c}_x &= \sum_{y=0}^{N-1} c_y e^{2\pi i x y/N} \\ &= \sum_{y=0}^{r-1} \left(c_y e^{2\pi i x y/N} + c_{y+r} e^{2\pi i x (y+r)/N} + c_{y+2r} e^{2\pi i x (y+2r)/N} + \cdots + c_{y+[m-1]r} e^{2\pi i x \left(y+[m-1]r\right)/N} \right) \\ &= \sum_{y=0}^{r-1} c_y e^{2\pi i x y/N} \left(1 + e^{2\pi i x/m} + e^{4\pi i x/m} + \cdots + e^{2(m-1)\pi i x/m} \right) \qquad \text{(Recall that } m = N/r\text{)} \\ &= \begin{cases} m \displaystyle\sum_{y=0}^{r-1} c_y e^{2\pi i x y/N} & x = \text{ (integer)} \times m \\ 0 & \text{otherwise} \end{cases},\end{aligned} \tag{6.5.7}$$

where the last result follows from (6.5.3), which reduces the sum in parentheses in the line above to m or 0. If indeed $x = km$, with k an integer, then $e^{2\pi i x y/N} = e^{2\pi i k y/r}$ has a period r/k in y that is commensurate with the period of c_y, so that the same number is created by the Fourier transform for each set of r terms in the sum, and the results added constructively. But for other values of x the function c_y and the exponential are out of phase and the Fourier transform creates (and then adds together) different numbers, with different phases, during each period of c_y. The result is destructive interference. The upshot of all this is that if when you Fourier-transform a periodic function, you find, provided that the N is an integer multiple of the period, that you have nonzero values only when x is a multiple of $m = N/r$. Seems like a good way to determine r.

Let's look at the example we'll be most concerned with:

$$c_y = \begin{cases} 1 & y = 0, r, 2r, \ldots, mr = N \\ 0 & \text{elsewhere} \end{cases}. \tag{6.5.8}$$

In this case, the only c between c_0 and c_{r-1} that is nonzero is c_0, so we have

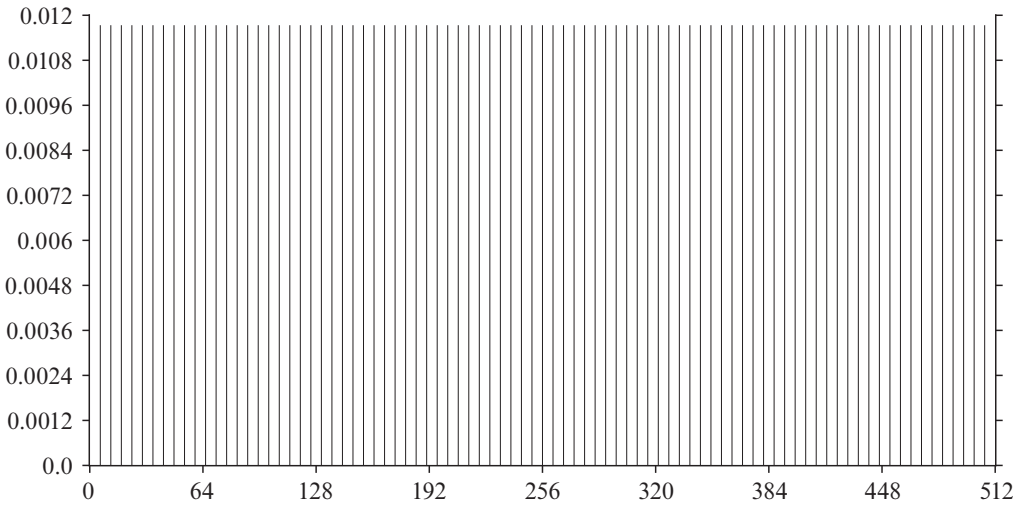
$$\sum_{y=0}^{r-1} c_y e^{2\pi i x y/N} = c_0 e^{2\pi i \cdot 0 \cdot y/N} = 1\,. \tag{6.5.9}$$

Eq. (6.5.7) then implies that

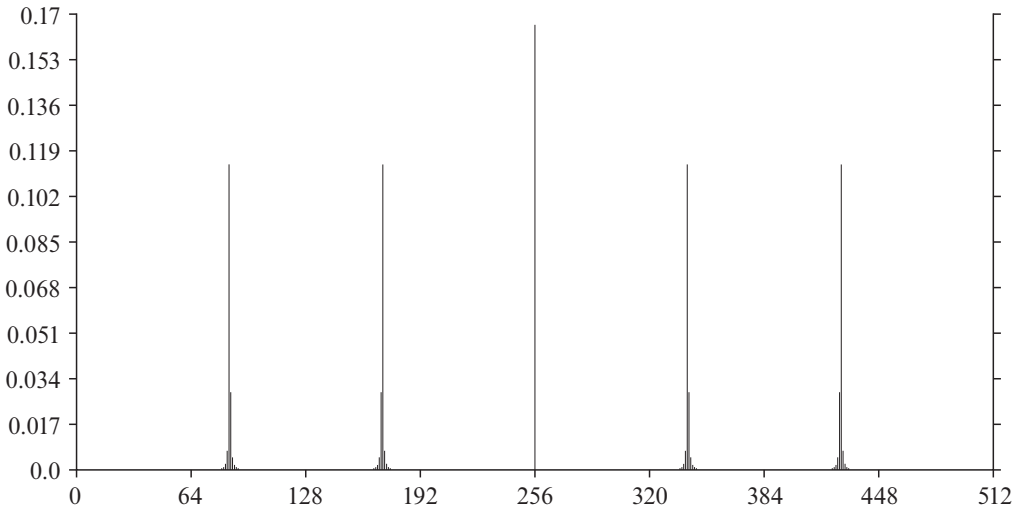
$$\tilde{c}_x = \begin{cases} m & x = \text{(integer)} \times m \\ 0 & \text{otherwise} \end{cases}\,. \tag{6.5.10}$$

The Fourier transform has spikes at values of x that are multiples of $m \equiv N/r$.

What happens if the spikes don't fit evenly into our set of numbers, that is for periods r if N/r is not an integer? Let's take a brief look and analyze it more carefully later. Rieffel and Polak's figure below shows a set of spikes with a period that does *not* evenly divide N:



The next figure shows the resulting square of the Fourier transform (because the square is what will matter in the quantum computer) In this last figure there are smaller peaks near the large ones at multiples of m. Their presence forced Shor to go through some contortions to extract the period r from the Fourier transform, as we'll see.



6.6 Using a Quantum Computer to Find Periods

Let n_0 be the number of bits such that 2^{n_0} is the smallest power of 2 that exceeds $M = pq$. If $M \approx 10^{500}$, then $n_0 \approx 1700$. To find the period of a function $f(x)$ (which in our case will be $a^x \mod N$ for some integer a of our choice but can be more general) by random searching you'd need to evaluate f a number of times that grows exponentially with n_0. Quantum computing, as we'll see, allows a number that grows roughly quadratically.

The "quantum core" as the Rieffel and Polak call the quantum part of Shor's algorithm, starts with the usual application of W and U_f to obtain

$$|\Psi\rangle = U_f\,(W \otimes I)\,|0\rangle_n\,|0\rangle_{n_0} = \sum_{x=0}^{N-1} |x\rangle_n\,|f(x)\rangle_{n_0}\,, \tag{6.6.1}$$

where n is the number of bits in our quantum computer (with $N = 2^n$) and n_0, as above, is the number you need to express M the large product of primes that we want to factor. We will need to choose

$$n = 2n_0\,, \tag{6.6.2}$$

for reasons we will see shortly.

Now suppose that, as in Simon's algorithm, we measure the value of the output register. Then (also like in Simon's problem) $|\Psi\rangle$ will collapse to one of may possible states of the form

$$|\Phi\rangle = \frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} |\overline{x} + kr\rangle\,|f(\overline{x})\rangle\,, \tag{6.6.3}$$

for some random $\overline{x}$. As before, I've dropped the subscripts on the kets that indicate the number of qubits. Here $\overline{x}$ is the smallest number for which $f(\overline{x})$ is the value that's measured, and m is the smallest integer for which $\overline{x} + mr \geq N$. Thus

$$m = \left[\frac{N}{r}\right] \quad \text{or} \quad \left[\frac{N}{r}\right] + 1\,, \tag{6.6.4}$$

depending on the value of $\overline{x}$. For example, if $N = 4$ and $r = 3$, then $\overline{x}$ must be 0, 1, or 2, because for $f(3) = f(0)$ and so three is not the smallest number that gives $f(x) = f(3)$. If $\overline{x} = 1$ or 2, then then $m = 0 = [4/3]$. But if $\overline{x} = 0$ then $m = 1$. Do you see why?

We'd like to get r from $|\Phi\rangle$, but just as in Simon's problem, the random number $\overline{x}$ prevents us from getting any information about it by measuring the input. We need an additional transformation that puts $\overline{x}$ into a phase. And the something based on a discrete Fourier transform seems like a good idea.

6.6.1 Quantum Fourier Transform

Here's the transformation that we need:

$$U_{FT}|x\rangle = \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{2\pi i x y/N} |y\rangle \tag{6.6.5}$$

U_{FT} is unitary because, letting $|FT_x\rangle \equiv U_{FT}|x\rangle$, we get

$$\begin{aligned}
\langle FT_x | FT_{x'} \rangle &= \frac{1}{N} \sum_{y,y'=0}^{N-1} \langle y|y'\rangle \, e^{-2\pi i x y/N} e^{2\pi i x' y'/N} \\
&= \frac{1}{N} \sum_{y=0}^{N-1} e^{2\pi i (x'-x) y/N} \\
&= \text{(from (6.5.3))} \begin{cases} 1 & x = x' \qquad (x - x' \text{ is too small to be a nonzero multiple of } N.) \\ 0 & \text{otherwise} \end{cases} .
\end{aligned} \tag{6.6.6}$$

So U_{FT} doesn't change inner products among a complete set of basis states. We'll see soon that it's possible to implement U_{FT} in terms of a small number of one- and two-qubit gates only! That's what makes the Shor algorithm so fast.

What happens when you apply U_{FT} to a superposition (we'll eventually want to apply it to the input of (6.6.3))? You get

$$\begin{aligned}
U_{FT}\left(\sum_{x=0}^{N-1} c_x |x\rangle\right) &= \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} c_x \sum_{y=0}^{N-1} e^{2\pi i x y/N} |y\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \underbrace{\left(\sum_{x=0}^{N-1} c_x e^{2\pi i x y/N}\right)}_{\tilde{c}_y} |y\rangle \\
&= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \tilde{c}_y |y\rangle
\end{aligned} \tag{6.6.7}$$

So when U_{FT} acts on a superposition, it gives a another superposition with coefficients that are the discrete Fourier transform of the original set of coefficients.

U_{FT} is a straightforward generalization of the Walsh-Hadamard transformation W.

Recall that the action of W can be written as

$$\begin{aligned} W|x\rangle &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} (-1)^{x\cdot y} |y\rangle \\ &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} e^{i\pi x\cdot y} |y\rangle \,. \end{aligned} \tag{6.6.8}$$

For a single qubit ($N = 2$) we have

$$\begin{aligned} U_{FT}|x\rangle &= \frac{1}{\sqrt{2}} \sum_{y=0}^{1} e^{2i\pi ixy/2} |y\rangle \\ &= \frac{1}{\sqrt{2}} \sum_{y=0}^{1} e^{i\pi xy} |y\rangle \\ &= \frac{1}{\sqrt{2}} \sum_{y=0}^{1} (-1)^{xy} |y\rangle \\ &= H|x\rangle \,. \end{aligned} \tag{6.6.9}$$

That is, for one qubit, $U_{FT} = H$.

OK, how do we use the quantum Fourier transform to extract r from (6.6.3)? Well, because the input and output are not entangled in $|\Phi\rangle$ from that equation (because $f(\overline{x})$ has only one value), we can write $|\Phi\rangle$ as

$$\begin{aligned} |\Phi\rangle &= |\Phi_{\text{in}}\rangle\, |f(\overline{x})\rangle \\ |\Phi_{\text{in}}\rangle &= \frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} |\overline{x} + kr\rangle \,. \end{aligned} \tag{6.6.10}$$

The coefficients are a series of equally spaced spikes, the distribution we looked at earlier. Applying U_{FT} to $|\Phi_{\text{in}}\rangle$ gives

$$\begin{aligned} U_{FT}|\Phi_{\text{in}}\rangle &= \frac{1}{\sqrt{N}} \sum_{y=0}^{N-1} \frac{1}{\sqrt{m}} \sum_{k=0}^{m-1} e^{2\pi i(\overline{x}+kr)y/N} |y\rangle \\ &= \sum_{y=0}^{N-1} e^{2\pi i\overline{x}y/N} \frac{1}{\sqrt{Nm}} \left(\sum_{k=0}^{m-1} e^{2\pi ikry/N} \right) |y\rangle \,. \end{aligned} \tag{6.6.11}$$

Now $\overline{x}$ is in a phase, and the probability of measuring the input register to be in the state $|y\rangle$ is independent of it:

$$\begin{aligned} P(y) &= \frac{1}{Nm} \left| e^{2\pi i\overline{x}y/N} \right|^2 \left| \sum_{k=0}^{m-1} e^{2\pi ikry/N} \right|^2 \\ &= \frac{1}{Nm} \left| \sum_{k=0}^{m-1} e^{2\pi ikry/N} \right|^2 . \end{aligned} \tag{6.6.12}$$

If we got incredibly lucky and N was a multiple of r, so that our register contained $m = N/r$ complete periods, then, as we showed in (6.5.10), $P(y) \propto \sum_{k=0}^{m-1} \exp(2\pi iy/m)$ would

be zero except at integer multiples of m, and running the quantum Fourier transform a few times would allow us to extract m and thus r. We won't usually be that lucky, though, and we'll need to figure out what to do when the probability distribution is peaked at values of y that are multiples of m, but not strictly zero at other values of y. That's next.

6.6.2 Extracting the Period

As we just said, if N is not an integer multiple of the period r, then there's a chance that we will measure a value of y that is not quite a multiple of m. But will show here that as long as $N > M^2$, where M is the large product of two primes that we want to factor, the chances of measuring a value of y that is within $\pm\frac{1}{2}$ of an integer multiple j of N/r, is greater than 40%. And if it is, one can use the continued fraction expansion of y to extract jN/r. You're not responsible for knowing what's in this little section this semester, but I've left it in the notes in case you're curious. It's all discussed in more detail in Mermin.

How do we show the 40% thing? Let's look at $P(y)$ for

$$\begin{aligned} y &= j\frac{N}{r} + \delta_j \\ &\equiv y_j \quad \text{with } j \text{ an integer and } |\delta_j| \le \frac{1}{2}. \end{aligned} \tag{6.6.13}$$

Here y_j is an integer (because it's one of the values of the register that we could measure) but jN/r will not be if N is not a multiple of the period r. From (6.6.12) we get

$$\begin{aligned} P(y_j) &= \frac{1}{Nm}\left|\sum_{k=0}^{m-1} e^{2\pi i k r(jN/r+\delta_j)/N}\right|^2 \\ &= \frac{1}{Nm}\left|\sum_{k=0}^{m-1} \underbrace{e^{2\pi i k j}}_{1} e^{2\pi i k r\delta_j/N}\right|^2 \\ &= \frac{1}{Nm}\sum_{k=0}^{m-1}\left|\left(e^{2\pi i r\delta_j/N}\right)^k\right|^2 \\ &= \frac{1}{Nm}\left|\frac{1-e^{2\pi i m r\delta_j/N}}{1-e^{2\pi i r\delta_j/N}}\right|^2. \end{aligned} \tag{6.6.14}$$

Using

$$\sin x = \frac{e^{ix}-e^{-ix}}{2}, \tag{6.6.15}$$

and factoring $\exp(\pi i m r\delta_j/N)$ out of the numerator (and $\exp(\pi i r\delta_j/N)$ out of the denominator), we get

$$P(y_j) = \frac{1}{Nm}\left[\frac{\sin\left(m\pi r\delta_j/N\right)}{\sin\left(\pi r\delta_j/N\right)}\right]^2 \tag{6.6.16}$$

At this point we start making approximations. From (6.6.4) m is within an integer of N/r, and $r << N$ (because $M << N$ and $r < M$) so

$$\begin{aligned} \frac{mr}{N} &= \frac{m}{N/r} \\ &= 1 + \mathcal{O}\left(\frac{r}{N}\right) \approx 1, \end{aligned} \tag{6.6.17}$$

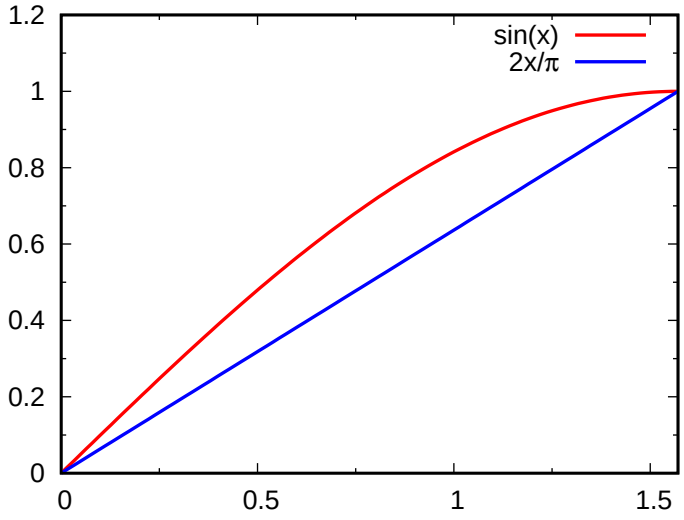
So we can replace replace $\pi\delta_j mr/N$ by $\pi\delta_j$ in the argument of the sin in the numerator of (6.6.16). Also, because r/N is very small, we can replace $\sin(\pi\delta_j r/N)$ by $\pi\delta_j r/N$ in the denominator of (6.6.16). The result is

$$\begin{aligned} P(y_j) &\approx \frac{1}{Nm}\left[\frac{\sin(\pi\delta_j)}{\pi\delta_j r/N}\right]^2 \\ &= \frac{N}{mr^2}\left[\frac{\sin(\pi\delta_j)}{\pi\delta_j}\right]^2 \\ &\approx \frac{1}{r}\left[\frac{\sin(\pi\delta_j)}{\pi\delta_j}\right]^2, \end{aligned} \tag{6.6.18}$$

where in the last step we've used the fact that $m \approx N/r$.

We're almost there now. The figure below shows that

$$\sin x \geq \frac{2x}{\pi} \quad \text{for } x \leq \frac{\pi}{2}. \tag{6.6.19}$$



So

$$\frac{\sin x}{x} \geq \frac{2}{\pi} \quad \text{for } x \leq \frac{\pi}{2}. \tag{6.6.20}$$

Since $|\delta_j| \leq 1/2$, we know that $|\pi\delta_j| \leq \pi/2$ and we have

$$P(y_j) \geq \frac{1}{r}\left(\frac{2}{\pi}\right)^2 = \left(\frac{4}{\pi^2}\right)\frac{1}{r}. \tag{6.6.21}$$

Finally, (6.6.13) implies that there are about r possible values for j (because Nj/r can be anything between 0 and N), so the probability of being within $\delta_j = 1/2$ of one of them is about $r(4/\pi^2)(1/r) = 4/\pi^2 = .405\ldots$. **In other words, you have a more than 40% chance of measuring a value of *y* that is with 1/2 of *jN/r* for *some j*.**

How do we extract the value of jN/r? Well, first we need to know that provided the integer y we measured really is within $1/2$ of such a number, that number is unique, i.e.

that our y is within $1/2$ of only one multiple of N/r. What's the reason for that? Well, we can write our "within $1/2$" property

$$|y_j - jm| < \frac{1}{2}, \tag{6.6.22}$$

as

$$\left|\frac{y_j}{N} - \frac{j}{r}\right| < \frac{1}{2N} \quad < \frac{1}{2M^2}, \tag{6.6.23}$$

where the last line comes from our making sure that N is at least M^2 (and this is the *real* reason we have to make M that big). We know that $r < M$ and that any two numbers with denominator less than M differ by more than $1/M^2$. That last thing is because

$$\begin{aligned} \left|\frac{a}{M_1} - \frac{b}{M_2}\right| &= \left|\frac{M_2 a - M_1 c}{M_1 M_2}\right| \\ &\geq \frac{1}{M_1 M_2}, \end{aligned} \tag{6.6.24}$$

so that if M_1 and M_2 are both at most M, then any two fractions with those numbers as denominators must differ by more than $1/M^2$. Thus, if our y_j is within $1/2$ of jN/r for some integer j, then because any two numbers of the form j/r are further apart by more than $1/M^2$, only one of them can be within $1/2M^2$ of our y_j/N. Or, in other words, at most one value of jN/r differs from our y_j by less than $1/2$. One can then find that unique value — actually, the unique value of j/r — by using the the continued-fraction expansion of number y_j/N. I won't go into how that works but will just quote the relevant theorem:

> If j is an integer and
>
> $$\left|x - \frac{j}{r}\right| < \frac{1}{2r^2},$$
>
> then j/r is one of the partial sums in the continued fraction expansion of x.

In our case, $x = y_j/N$. Because $M > r$, Eq. (6.6.23) implies that y_j/N is indeed closer than $1/(2r^2)$ to j/r for some j, and you can identify j/r by using continued fractions. Even after you extract j/r, however, you still don't quite have r. You may have to run the quantum part of the algorithm a few times to get it for sure, and deal with the fact that your measured y won't always be within $1/2$ of a multiple of N/r. I won't say how you isolate r to very high probability, but will simply tell you that these extra steps don't add much computation time. And so, that's Shor's algorithm. 🙂

6.7 Implementation of Quantum Fourier Transform

Here we'll show that U_{FT} can be implemented "spectacularly efficiently," as Mermin says, with a small number of one- and two-qubit gates. To make things a little less abstract, I'll consider a four-qubit system. It will be easy to see how the implementation generalizes to more qubits.

Let's write our four-qubit states explicitly as

$$|y\rangle = |y_3 y_2 y_1 y_0\rangle \tag{6.7.1}$$

with

$$y = 8y_3 + 4y_2 + 2y_1 + y_0\,. \tag{6.7.2}$$

This allows us to manipulate the action of U_{FT} on an arbitrary standard-basis state $|x\rangle$ as follows:

$$\begin{aligned}
U_{FT}|x\rangle &= \frac{1}{4}\sum_{y=0}^{15} e^{2\pi i x y/16}|y\rangle \\
&= \frac{1}{4}\sum_{y=0}^{15} e^{2\pi i x(y_0+2y_1+4y_2+8y_3)/16}|y\rangle \\
&= \frac{1}{4}\sum_{y_3=0}^{1}\sum_{y_2=0}^{1}\sum_{y_1=0}^{1}\sum_{y_0=0}^{1} e^{i\pi x y_3}|y_3\rangle \otimes e^{i\pi x y_2/2}|y_2\rangle \otimes e^{i\pi x y_1/4}|y_1\rangle \otimes e^{i\pi x y_0/8}|y_0\rangle \\
&= \frac{1}{4}\sum_{y_3=0}^{1} e^{i\pi x y_3}|y_3\rangle \otimes \sum_{y_2=0}^{1} e^{i\pi x y_2/2}|y_2\rangle \otimes \sum_{y_1=0}^{1} e^{i\pi x y_1/4}|y_1\rangle \otimes \sum_{y_0=0}^{1} e^{i\pi x y_0/8}|y_0\rangle \\
&= \frac{1}{4}\left(|0\rangle + e^{i\pi x}|1\rangle\right) \otimes \left(|0\rangle + e^{i\pi x/2}|1\rangle\right) \otimes \left(|0\rangle + e^{i\pi x/4}|1\rangle\right) \otimes \left(|0\rangle + e^{i\pi x/8}|1\rangle\right)
\end{aligned} \tag{6.7.3}$$

Now we note that, because $x = 2^3 x_3 + 2^2 x_2 + 2x_1 + x_0$,

$$x = 2 \times \text{ integer } + x_0\,. \tag{6.7.4}$$

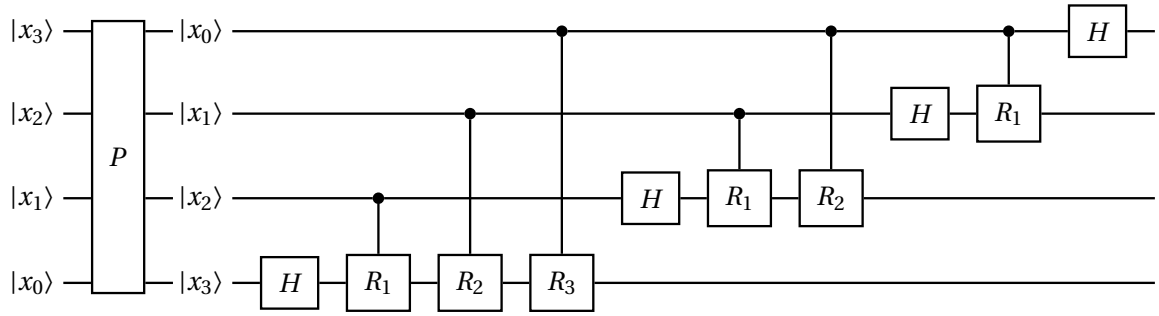
Similarly

$$\begin{aligned}
\frac{x}{2} &= 2 \times \text{ integer } + x_1 + \frac{x_0}{2} \\
\frac{x}{4} &= 2 \times \text{ integer } + x_2 + \frac{x_1}{2} + \frac{x_0}{4}\,.
\end{aligned} \tag{6.7.5}$$

When we raise e to a power $2k\pi i$, with k an integer, we just get 1. As a result, we have

$$\begin{aligned}
U_{FT}|x\rangle = \frac{1}{4}&\left(|0\rangle + e^{i\pi x_0}|1\rangle\right) \otimes \left(|0\rangle + e^{i\pi(x_1+x_0/2)}|1\rangle\right) \\
&\otimes \left(|0\rangle + e^{i\pi(x_2+x_1/2+x_0/4)}|1\rangle\right) \otimes \left(|0\rangle + e^{i\pi(x_3+x_2/2+x_1/4+x_0/8)}|1\rangle\right)
\end{aligned} \tag{6.7.6}$$

Now, I claim that the circuit that implements U_{FT} looks like this:



Here P is a transformation built out of swaps to invert the order of the qubits and R_k is the transformation

$$\begin{aligned} R_k|0\rangle &= 0 \\ R_k|1\rangle &= e^{i\pi/2^k}. \end{aligned} \tag{6.7.7}$$

Why? Well, because $\exp(i\pi x)$ is 1 if $x = 0$ and -1 if $x = 1$, applying the first Hadamard to $|x_3\rangle$ produces the state

$$|\Psi\rangle = I\otimes I\otimes I\otimes H|x_0x_1x_2x_3\rangle = \frac{1}{\sqrt{2}}|x_0x_1x_2\rangle\left(|0\rangle + e^{i\pi x_3}|1\rangle\right). \tag{6.7.8}$$

Then the controlled R_1 gate gives

$$\begin{aligned} I\otimes I\otimes C_{R_1}|\Psi\rangle &= \frac{1}{\sqrt{2}}|x_0x_1x_2\rangle\left(|0\rangle + e^{i\pi x_3}e^{i\pi x_2/2}|1\rangle\right) \\ &= \frac{1}{\sqrt{2}}|x_0x_1x_2\rangle\left(|0\rangle + e^{i\pi(x_3+x_2/2)}|1\rangle\right). \end{aligned} \tag{6.7.9}$$

That's because the controlled R_1 doesn't change anything if $x_2 = 0$, but multiplies the piece of the state with the last qubit in $|1\rangle$ by $\exp(i\pi/2)$ if $x_2 = 1$. Then, after applying the controlled R_2 and R_3 gates to the last qubit, we convert our state to

$$\frac{1}{\sqrt{2}}|x_0x_1x_2\rangle\left(|0\rangle + e^{i\pi(x_3+x_2/2+x_1/4+x_0/8)}|1\rangle\right). \tag{6.7.10}$$

OK, that's pretty much it, as we can see by moving on to the preceding qubit, which is in the state x_2. The gates that act on it lead to

$$\frac{1}{\sqrt{2}}|x_0x_1\rangle\left(|0\rangle + e^{i\pi(x_2+x_1/2+x_0/4)}|1\rangle\right)\left(|0\rangle + e^{i\pi(x_3+x_2/2+x_1/4+x_0/8)}|1\rangle\right), \tag{6.7.11}$$

in the same way as we've just seen, and when we add in those that act on the preceding two qubits we recover (6.7.6). And it's clear how this circuit generalizes to more qubits.

The upshot is, as I claimed, that only one- and two-qubit gates, and not very many of them, appear in the circuit that implements U_{FT}. How many of them appear in the n-qubit case? Well, we clearly then have $N_1 = n$ one-qubit gates and

$$\begin{aligned} N_2 &= 1 + 2 + \cdots + n - 1 \\ &= \frac{n(n-1)}{2} \end{aligned} \tag{6.7.12}$$

two-qubit gates. So the number of gates is quadratic in the number of bits, and that is the more or less how the entire Shor algorithm scales. No "classical" algorithm has been found that is faster than exponential in the number of bits. A quadratic scaling would make RSA encryption obsolete.

I haven't had time to cover everything needed to rigorously prove that this algorithm scales quadratically (or, with improvements, even better) with the number of bits. The most important thing we didn't discuss, and it would be pretty easy to do so, is efficient modular exponentiation. That's needed to construct the function $a^x \mod M$ that U_{FT} finds the period of. We just don't have quite enough time for it.

7 Grover's Algorithm

7.1 The Problem

Grover's algorithm does "unstructured search" for an item having a particular property. What is that? Well, it's not trying, for example, to find a phone number for a particular person in a file of matched names and numbers. The people's names are generally ordered in the file (alphabetically) and so it's easy to find the number of any person. It's more like the problem of trying to find a person who has a particular number. With a regular computer you have to check numbers one at a time until you find the one you're interested in. If you have N numbers in the file you'd have to "query" the database $N/2$ times to have a 50% chance of finding the person you're after. But with a quantum database — one that you could store in "quantum memory" that is capable of putting items in superposition — you'd have to query only $\sqrt{N}$ times to have a very good chance of finding the person. You don't get an exponential speedup like in Shor's algorithm, but it's still a speedup provided that U_f can be efficiently implemented.

We can formalize this as another oracle problem. Our oracle evaluates a function for which

$$f(x) = \begin{cases} 0\,, & x \neq a \\ 1\,, & x = a \end{cases}\,. \tag{7.1.1}$$

for some one number a. The problem is to find a.

7.2 Amplitude Amplification

The first step of the algorithm, as was the case several times before, will be to act with $W \otimes (HX)$ on $|0\rangle\,|0\rangle$, where the input register contains n (many) qubits but the output only one qubit:

$$\begin{aligned} W \otimes (HX)\,|0\rangle\,|0\rangle &= \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle\,|-\rangle \\ &\equiv |\Phi\rangle\,|-\rangle\,, \end{aligned} \tag{7.2.1}$$

with $N = 2^n$ and

$$|\Phi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle\,. \tag{7.2.2}$$

The next step is to apply U_f for the function given in Eq. (7.1.1). (We only need one output qubit because the only values of that function are 0 and 1). Recall that

$$U_f\,|x\rangle\,|-\rangle = (-1)^{f(x)}\,|x\rangle\,|-\rangle\,. \tag{7.2.3}$$

The state isn't changed except for a sign, so we can write

$$U_f\,|x\rangle\,|-\rangle = V_f \otimes I\,|x\rangle\,|-\rangle\,, \tag{7.2.4}$$

and just concern ourselves with V_f:

$$V_f\,|x\rangle = (-1)^{f(x)}\,|x\rangle = \begin{cases} \;\;|x\rangle\,, & x \neq a\,. \\ -|a\rangle\,, & x = a\,. \end{cases} \tag{7.2.5}$$

If we have a superposition

$$|\Psi\rangle \equiv c_0|0\rangle + c_1|1\rangle + \cdots + c_a|a\rangle + \ldots c_{N-1}|N-1\rangle\,, \tag{7.2.6}$$

then

$$\begin{aligned} V_f|\Psi\rangle &= c_0|0\rangle + c_1|1\rangle + \cdots - c_a|a\rangle + \ldots \\ &= |\Psi\rangle - 2c_a|a\rangle \end{aligned} \tag{7.2.7}$$

Recalling that $c_a = \langle a|\Psi\rangle$, we can write

$$V_f|\Psi\rangle = |\Psi\rangle - 2|a\rangle\langle a|\Psi\rangle\,, \tag{7.2.8}$$

so that

$$V_f = I - 2|a\rangle\langle a|\,. \tag{7.2.9}$$

No matter how we write this transformation, its effect is to reverse the sign of the component along $|a\rangle$.

Now let's define another transformation. For a general state $|\Psi\rangle$ we let

$$\overline{c} = \frac{1}{N}\sum_{k=0}^{N-1} c_k\,, \tag{7.2.10}$$

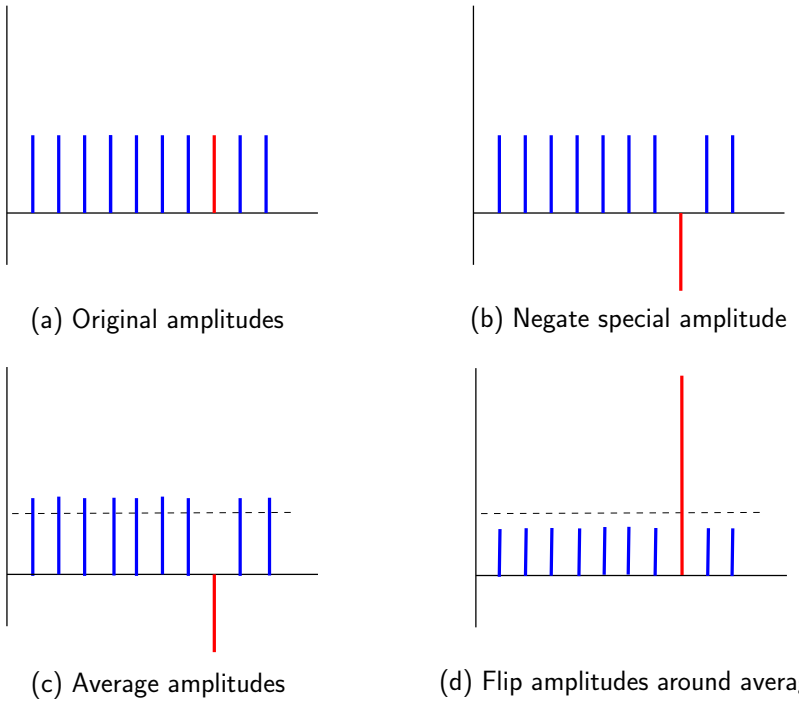
be the "average amplitude" in $|\Psi\rangle$, and define

$$S|\Psi\rangle = 2\overline{c}\,(|0\rangle + \cdots + |N-1\rangle) - (c_0|0\rangle + \cdots + c_{N-1}|N-1\rangle) \tag{7.2.11}$$

This transformation replaces all the amplitudes c_k in $|\Psi\rangle$ with $2\overline{c} - c_k$, which means that

$$\begin{aligned} c_k &\longrightarrow 2\overline{c} - c_k \\ \Longrightarrow \quad c_k - \overline{c} &\longrightarrow 2\overline{c} - c_k - \overline{c} \\ &= \overline{c} - c_k\,. \end{aligned} \tag{7.2.12}$$

In words, this means that the signs of the coefficients are changed "with respect to the average amplitude." The figure below shows what happens when SV_f acts on the uniform superposition $|\Phi\rangle$, which is the initial state of the input in Eq. (7.2.1). V_f acts first, negating the amplitude corresponding to state $|a\rangle$ (the red line). Then S acts, flipping the amplitudes around their average value, which is given by the dotted line.



(a) Original amplitudes

(b) Negate special amplitude

(c) Average amplitudes

(d) Flip amplitudes around average

The amplitude for $|a\rangle$ gets amplified and all the rest shrink. If we apply SV_f again, and then again, etc. (but not too many, it turns out), the amplitude we want will be much larger than all the rest, and if we measure the value of the input register we'll obtain a with high probability.

We can write a simple expression for S by noting that

$$\begin{aligned} \overline{c} &= \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} \frac{1}{\sqrt{N}} c_k \\ &= \frac{1}{\sqrt{N}} \langle \Phi | \Psi \rangle \,, \end{aligned} \tag{7.2.13}$$

so that

$$\begin{aligned} S|\Psi\rangle &= 2\overline{c}\sqrt{N}|\Phi\rangle - |\Psi\rangle \\ &= 2|\Phi\rangle\langle\Phi|\Psi\rangle - |\Psi\rangle \,, \end{aligned} \tag{7.2.14}$$

which means that

$$S = 2|\Phi\rangle\langle\Phi| - I \tag{7.2.15}$$

7.3 Geometric Interpretation

We could use the fact that repeated application of SV_f to $|\Phi\rangle$ kills amplifies the amplitude c_a while killing off all the rest, at least for a while, to figure out the best number of times to repeat the application. But we're going to visualize the process from a geometric point of view instead.

Let's look at the action of V_f and S on $|a\rangle$ and $|\Phi\rangle$:

$$\begin{aligned} V_f|a\rangle &= (I - 2|a\rangle\langle a|)\ |a\rangle \\ &= -|a\rangle \end{aligned}$$

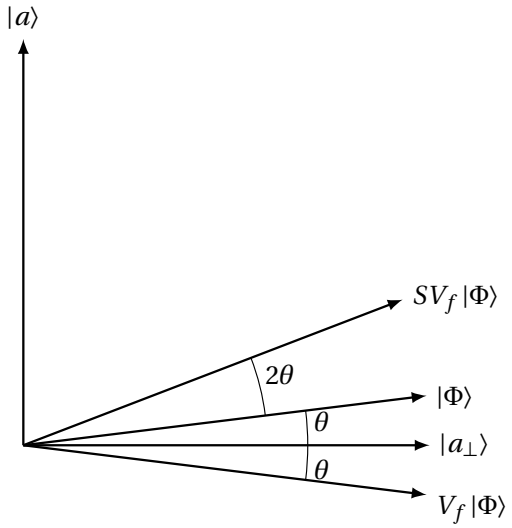
$$\begin{aligned} V_f|\Phi\rangle &= (I - 2|a\rangle\langle a|)\ |\Phi\rangle \\ &= |\Phi\rangle - 2|a\rangle \underbrace{\langle a|\Phi\rangle}_{1/\sqrt{N}} \\ &= |\Phi\rangle - \frac{2}{\sqrt{N}}|a\rangle\ , \end{aligned} \tag{7.3.1}$$

and[1]

$$\begin{aligned} S|a\rangle &= (2|\Phi\rangle\langle\Phi| - I)\ |a\rangle \\ &= 2|\Phi\rangle\ \langle\Phi|a\rangle - |a\rangle \\ &= \frac{2}{\sqrt{N}}|\Phi\rangle - |a\rangle \end{aligned} \tag{7.3.2}$$

$$\begin{aligned} S|\Phi\rangle &= (2|\Phi\rangle\langle\Phi| - I)|\Phi\rangle \\ &= |\Phi\rangle\ . \end{aligned}$$

Thus these two operators, acting on any real linear combination of $|a\rangle$ and $|\Phi\rangle$ return other real combinations of those vectors, and if we are considering only such combinations we can restrict our attention to the real two-dimensional subspace spanned by $|a\rangle$ and $|\Phi\rangle$. The figure below depicts some vectors in that subspace. The vector $|a_\perp\rangle$ is just the unit vector perpendicular to $|a\rangle$ in this space.



Now we can look in a pictorial way at what V_f and S do. From the action in (7.2.5) or the more compact expression (7.2.9) we can see that V_f reverses the direction of $|a\rangle$

[1] Why is $\langle a|\Phi\rangle$ equal to $\frac{1}{\sqrt{N}}$?

and does nothing to any state that is orthogonal to $|a\rangle$. In other words, it reflects any state vector across the line parallel to $|a_\perp\rangle$. So, for example, in the figure, $V_f|\Phi\rangle$ is just the original vector Φ reflected across the horizontal vector $|a_\perp\rangle$. Similarly, from (7.2.15) we can see that S does nothing to $|\Phi\rangle$ (that's shown explicitly in (7.3.2)) and reverses the direction of any vector that's perpendicular to $|\Phi\rangle$. It thus reflects through the line parallel to $|\Phi\rangle$.

What happens when we apply SV_f to $|\Phi\rangle$? Well, if $|\Phi\rangle$ makes an angle θ from above with $|a_\perp\rangle$, then $V_f|\Phi\rangle$ makes the same angle from below, so that it makes an angle 2θ with $|\Phi\rangle$ itself. Then $SV_f|\Phi\rangle$ makes the same angle 2θ but from above with $|\Phi\rangle$. The net result of the combined action of S and V_f is to rotate $|\Phi\rangle$ by 2θ.

What if we now apply SV_f again to this rotated vector? Well, V_f will cause it to make an angle 3θ with $|a_\perp\rangle$ from below (and so 4θ with $|\Phi\rangle$ from below) and S will leave it making an angle 4θ from above with $|\Phi\rangle$. *So every time we apply the combination SV_f, we rotate the vector an additional angle 2θ counterclockwise.* To find $|a\rangle$, we just rotate $|\Phi\rangle$ enough times so that our state is as close as possible to $|a\rangle$?

How many times is that and how can we know how many without knowing $|a\rangle$? Well, although we don't know $|a\rangle$, we do know that $\langle a|\Phi\rangle = \frac{1}{\sqrt{N}}$. And since we are in a real space, we know that $\langle a|\Phi\rangle$ is the cosine of the angle between $|a\rangle$ and $|\Phi\rangle$, so that

$$\begin{aligned} \cos\left(\frac{\pi}{2}-\theta\right) &= \frac{1}{\sqrt{N}} \\ \Longrightarrow \quad \sin\theta &= \frac{1}{\sqrt{N}}. \end{aligned} \tag{7.3.3}$$

So we can figure out how many times we have to rotate $|\Phi\rangle$ by 2θ to get as close as possible to $90°$, starting from θ.

Let's look at the case we're most interested in, the limit in which N is really large. Then

$$\theta \approx \sin\theta = \frac{1}{\sqrt{N}}. \tag{7.3.4}$$

Then to get to $90°$ in k rotations, we require that

$$\begin{aligned} \frac{1}{\sqrt{N}} + k\cdot\frac{2}{\sqrt{N}} &= \frac{\pi}{2} \\ \Longrightarrow \quad k &= \frac{\frac{\sqrt{N}\pi}{2}-1}{2} \\ &\approx \frac{\sqrt{N}\pi}{4} \qquad (\text{since } N>1). \end{aligned} \tag{7.3.5}$$

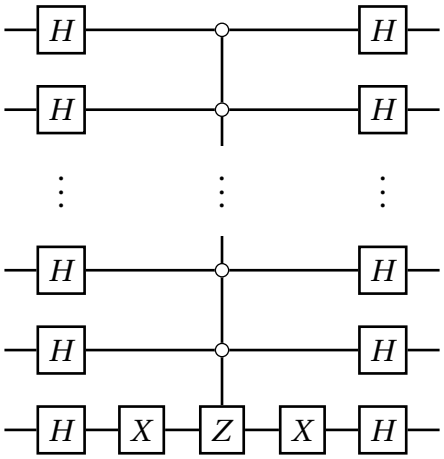
If we apply SV_f as close as possible to this many times (i.e by selecting the integer that is as close as possible to $\sqrt{N}\pi/4$), our state will be as close as possible to $|a\rangle$ and when we measure we will obtain a with high probability (the square of the cosine of the angle between or final vector and $|a\rangle$, or equivalently the square of the sine of the angle between our final vector and $|a_\perp\rangle$). We can check to make sure that we got the right answer by asking the oracle to evaluate the function at what we think is a. If it's not 1 we can repeat the procedure. Whether we repeat or not, the number of times we have to call the oracle is proportional to $\sqrt{N}$.

7.4 Implementation of S

We're assuming that the computer can efficiently evaluate V_f; that's just a matter of addressing its "quantum memory." But for Grover's algorithm to have a chance to be useful, there must also be a simple implementation of $S = 2|\Phi\rangle\langle\Phi| - I$. Remembering that $|\Phi\rangle = W|0\rangle$ and that W is its own adjoint and inverse, we write S as

$$\begin{aligned} S &= 2W|0\rangle\langle 0|W - I \\ &= W(2|0\rangle\langle 0| - I)W, \end{aligned} \tag{7.4.1}$$

where $|0\rangle$ here is the n-qubit state $|00\ldots0\rangle$. The gates that represent this operator are supposed to do nothing (i.e. multiply by 1) if the state coming in is $|\Phi\rangle$ and multiply by -1 if it's orthogonal to $|\Phi\rangle$. Thus, the gates corresponding to the piece inside the W's (i.e. inside the parentheses) should do nothing if the state is $|0\rangle$ (i.e. if all the individual qubits are in the single qubit state $|0\rangle$) and produce a minus sign if any of the qubits are in the single qubit state $|1\rangle$. This is the circuit that does what we want up to an overall sign, that is, it multiplies by -1 if the state is $|\Phi\rangle$ and does nothing otherwise (which in both cases is the negative of what we want):



Why? Well the role of the H's is clear. The stuff in the middle acts only when the first $N-1$ bits are zeros. When they are, XZX acts on the last qubit. What does that transformation do? Well,

$$\begin{aligned} XZX|0\rangle &= XZ|1\rangle = -X|1\rangle = -|0\rangle \\ XZX|1\rangle &= XZ|0\rangle = X|0\rangle = |1\rangle\,. \end{aligned} \tag{7.4.2}$$

So it multiplies the state by -1 (which becomes a global phase) when the last qubit is in the state $|0\rangle$ and leaves it alone if it's in the state $|1\rangle$. So all the qubits have to be in the state $|0\rangle$ for the circuit to do anything. If any of the first $N-1$ are not, only $XX = I$ acts. If the last is not, then XZX does nothing.

Does it matter that the circuit does the opposite of what we want, i.e. that it implements $-S$ rather than S? Well, that means that the operation will reflect the vector it acts on across $|\Phi\rangle$ and then reverse its direction. But that doesn't affect the square of the inner product with $|a\rangle$, and when you act an even number of times with SV_f it doesn't matter at all if you substitute $-S$ for S. And in any event, we know that a global phase in

front of a quantum state such as $S|\Psi\rangle$ changes absolutely nothing in any measurement that follows.

8 Introduction to Error Correction

Real quantum computers will always lead to errors because they cannot be perfectly shielded from the surrounding environment. As a result, the computer's register becomes entangled with molecules in the environment and the register no longer has its own quantum state. There is an entire mathematical formalism designed to deal with this situation that we won't get into. Instead I'll try to discuss the correction of single-qubit errors in a (relatively) simple way.

8.1 Bit Flips

Let's start by pretending that the environment manifests itself simply as random flips of single qubits, like the kind that happen in conventional computers. Correcting other kinds of errors will involve generalizations of what we do to correct these bit-flip errors.

At first glance, it might seem impossible that such errors could be corrected if they happen in superpositions of computational basis states, because a measurement in that basis collapses the state vector and destroys information about the original superposition. But it can be done, through auxiliary qubits that are measured only after interacting with the register, without entanglement and in a way that preserves the state of the register itself.

In conventional computers bit-flip errors are corrected through what is called "redundant encoding." A bit with value 0 can be replaced by three bits, each a 0. If one of the three bits is flipped accidentally you can tell by seeing that you're left with two 0's and only a single 1. If bit flips are rare enough, that observation tells you that the 1 should be a 0 and you simply flip it back and proceed with whatever you were doing. Of course if two bits flip before you check, you'll make a mistake, but if you know how often bits flip you can check frequently enough to make that unlikely.

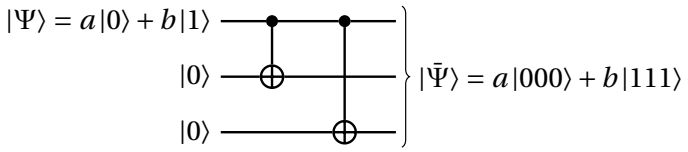
What about in a quantum computer? If all your states are computational-basis states, this scheme would work because nothing would be changed by measurement. But what if you have superpositions? Well, we can still do the encoding on qubit states:

$$\begin{aligned} |0\rangle &\longrightarrow |\bar{0}\rangle \equiv |000\rangle \\ |1\rangle &\longrightarrow |\bar{1}\rangle \equiv |111\rangle \, . \end{aligned} \tag{8.1.1}$$

Then the encoding will act on a superposition like so:

$$\begin{aligned} & |\Psi\rangle = a|0\rangle + b|1\rangle \\ \longrightarrow \quad & |\bar{\Psi}\rangle \equiv a|000\rangle + b|111\rangle \, . \end{aligned} \tag{8.1.2}$$

The transformation is easy to carry out, with this circuit:

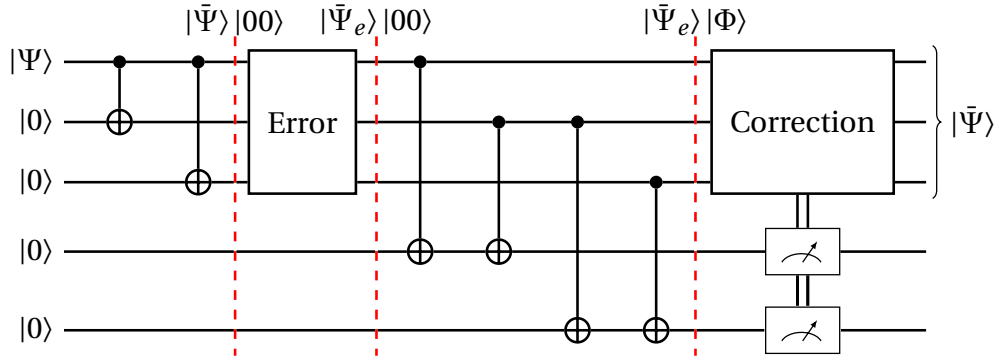


Now let us suppose that a random error can flip one of the qubits in some short time. Then after that time, the state $|\bar{\Psi}\rangle$ has been changed to one of the following (which include $|\bar{\Psi}\rangle$ itself because it's possible that no error occurs)

$$|\bar{\Psi}\rangle \longrightarrow |\bar{\Psi}_e\rangle = \begin{cases} a|000\rangle + b|111\rangle & \text{No bits flipped} \\ a|100\rangle + b|011\rangle & \text{Bit 1 flipped} \\ a|010\rangle + b|101\rangle & \text{Bit 2 flipped} \\ a|001\rangle + b|110\rangle & \text{Bit 3 flipped} \end{cases} . \tag{8.1.3}$$

To fix the error you have to identify which bit has been flipped (that's called the "syndrome," which more generally refers to the kind of error that has happened). And you have to identify the syndrome without learning anything about the coefficients a and b; if you did learn about them you'd cause the state to collapse, altering it irrevocably. How is it possible to identify the syndrome under such a restriction?

It turns out that you can do it with a couple of auxiliary bits. They make it possible to learn about what Mermin calls the "correlations among the qubits" without learning anything about a and b. If you can arrange for each of the four $|\bar{\Psi}_e\rangle$'s to put the two auxiliary qubits into a different state in an orthonormal basis, without entangling them with $|\bar{\Psi}_e\rangle$, then measuring the state of the auxiliary bits can tell you the syndrome. Indeed, consider what the circuit below does.



You can convince yourself that before the correction gate, the state $|\Phi\rangle$ of the two auxiliary qubits will depend on that of the top three in the following way:

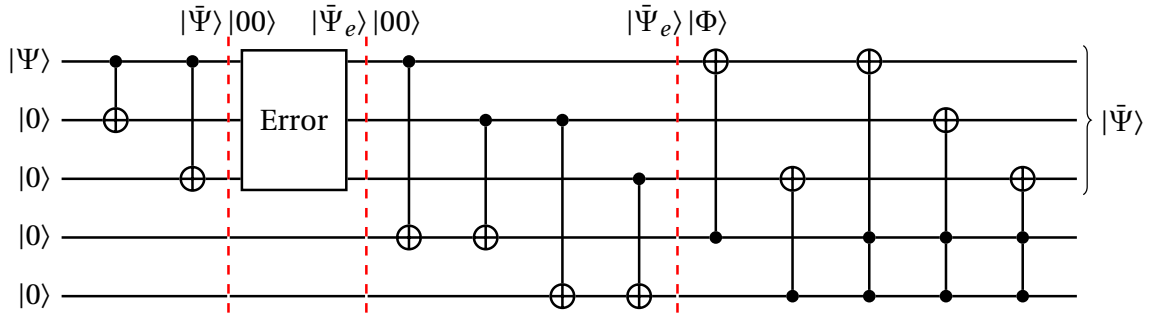
$\|\bar{\Psi}_e\rangle$	$\|\Phi\rangle$
$a\|000\rangle + b\|111\rangle$	$\|00\rangle$
$a\|100\rangle + b\|011\rangle$	$\|10\rangle$
$a\|010\rangle + b\|101\rangle$	$\|11\rangle$
$a\|001\rangle + b\|110\rangle$	$\|01\rangle$

Thus you can restore the original $|\bar{\Psi}\rangle$, after measuring the auxiliary qubits (the meter gates signify measurement) by applying corrections that depend on the results of the measurement, as in this table:

$\|\Phi\rangle$	correction to $\|\bar{\Psi}_e\rangle$
$\|00\rangle$	$I \otimes I \otimes I$
$\|01\rangle$	$I \otimes I \otimes X$
$\|10\rangle$	$X \otimes I \otimes I$
$\|11\rangle$	$I \otimes X \otimes I$

Ta dah!

As Mermin notes, this procedure can be "automated" so that measurement isn't necessary. You can check that the circuit below does the trick:



Finally, even if the bit flip error is a "small one", so that

$$|\bar{\Psi}\rangle|00\rangle \xrightarrow{\text{error}} \frac{1}{\sqrt{1+\varepsilon^2}}\left(|\bar{\Psi}\rangle + \varepsilon|\bar{\Psi}_e\rangle\right)|00\rangle\,, \tag{8.1.4}$$

for one of the altered states $|\bar{\Psi}\rangle_e$, the method still works. Suppose, for example $|\bar{\Psi}_e\rangle = a|100\rangle + b|011\rangle$. Then the error-identification part of the circuit, in which the two auxiliary qubits are altered by the controlled NOTs, produces the state

$$\frac{1}{\sqrt{1+\varepsilon^2}}\left(|\bar{\Psi}\rangle|00\rangle + \varepsilon|\bar{\Psi}_e\rangle|10\rangle\right). \tag{8.1.5}$$

The measurement of the auxiliary bits then collapses the encoded state into $|\bar{\Psi}\rangle$ if 00 is obtained (in which case you do nothing to the encoded state) or $|\bar{\Psi}_e\rangle$ if 10 is obtained (in which case you flip bit the first bit in the encoded state). If all four errors occur at some level, then Eq. (8.1.5) contains four terms, but the same idea applies.

8.2 Phase Errors

Bit flips are not the most general kind of single-qubit errors; we'll see how to represent general errors in the next section. For now, let's look at another kind of error, a "phase flip," in which the operator Z is inadvertently applied, so that $|0\rangle$ is unchanged but $|1\rangle$ is changed to $-|1\rangle$. With the encoding that we used to identify bit flips, a phase flip on any of the three qubits will cause the change:

$$|\bar{\Psi}\rangle = a|000\rangle + b|111\rangle \quad \longrightarrow \quad |\bar{\Psi}_e\rangle = a|000\rangle - b|111\rangle\,. \tag{8.2.1}$$

The circuit we constructed to diagnose and correct bit-flip errors will not detect the phase flip because $|\bar{\Psi}_e\rangle$ has the same form as $|\bar{\Psi}\rangle$, just with different coefficients. Is there another encoding that will allow phase flips to be detected?

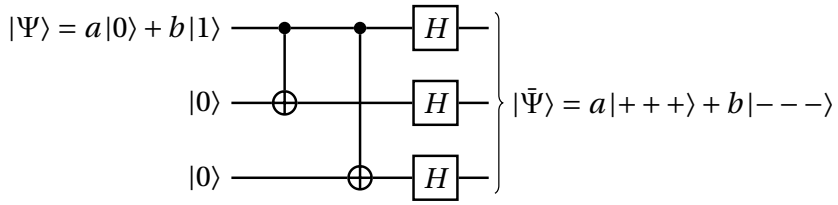
To see that there is, you just have to notice that

$$|+\rangle \xleftrightarrow{\text{phase flip}} |-\rangle\,, \tag{8.2.2}$$

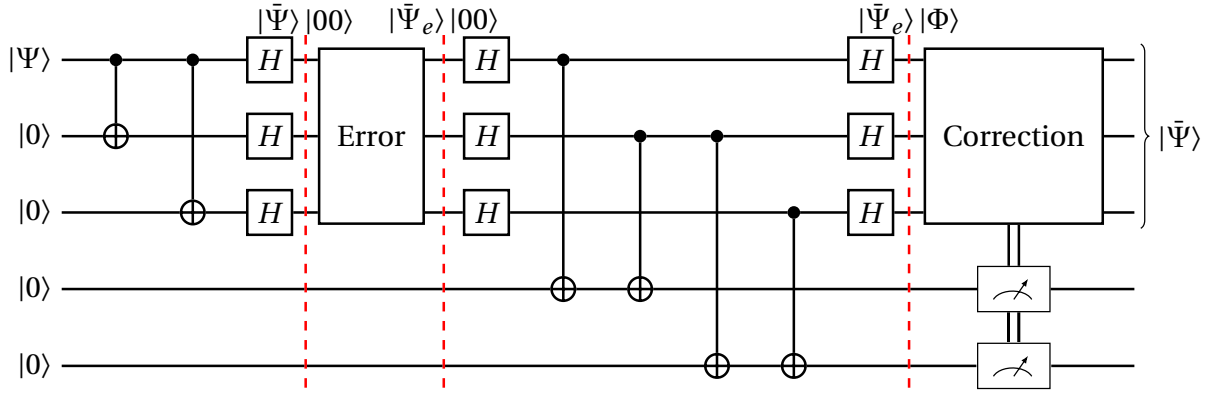
so that a phase flip interchanges $|+\rangle$ and $|-\rangle$ the same way that a bit flip interchanges $|0\rangle$ and $|1\rangle$. This parallel suggests the encoding

$$\begin{aligned} |\bar{0}\rangle &= |+++\rangle \\ |\bar{1}\rangle &= |---\rangle\,, \end{aligned} \tag{8.2.3}$$

a transformation created by this encoding circuit:



How do we identify and decode the error? This circuit does it:



Here the measurement is in the Hadamard basis. Note that this circuit, while it identifies and corrects phase errors, will not detect bit-flip errors, which are just like phase-flip errors in the original encoding.

Can we construct a circuit that will correct both kinds of errors? Before answering, let's address the question of the most general kind of single-qubit errors that we are likely to encounter.

8.3 The Form of General Single-Qubit Errors

To understand what happens more generally, we need to model the environment with which the qubits entangle (a process called "decoherence"). I will leave it to Mermin to convince you that the environment that will eventually become entangled with our qubit can be assigned a gigantic many particle state $|e\rangle$ that specifies the probabilities of

measuring any values in any experiments. He says that if you don't buy it, just imagine that the environment is a huge array of qubits, to which we know we can assign a state.

Initially, the environment and our error-prone qubit are not entangled, and so if the qubit initially is in a definite state in the computational basis, we can assign the joint environment-qubit system the state $|0\rangle|e\rangle$ or $|1\rangle|e\rangle$. When the environment and the qubit become entangled, the joint state changes:

$$\begin{aligned} |0\rangle|e\rangle &\longrightarrow |0\rangle|e_0\rangle + |1\rangle|e_1\rangle \\ |1\rangle|e\rangle &\longrightarrow |0\rangle|e_2\rangle + |1\rangle|e_3\rangle \,. \end{aligned} \tag{8.3.1}$$

Here $|e_0\rangle$, $|e_1\rangle$, $|e_2\rangle$, and $|e_3\rangle$, are other environmental states. In general, these states are *not normalized or orthogonal*, making the manipulation of the joint states a bit different from what we're used to. We are assuming that the decoherence is slow, so after a little while we would find that

$$\begin{aligned} |e_0\rangle \approx |e_3\rangle \approx |e\rangle \\ \langle e_1|e_1\rangle \ll 1 \quad \langle e_2|e_2\rangle \ll 1 \,. \end{aligned} \tag{8.3.2}$$

This means that the states on the left of (8.3.1) change only a little; the inner products being small in (8.3.2) is equivalent to $|e_1\rangle$ and $|e_2\rangle$ being normalized in (8.3.2) but having small coefficients in front of them.

It will be useful to combine the two equations in (8.3.1) into one by using the one-qubit projection operators $P_0 \equiv |0\rangle\langle 0|$ and $P_1 \equiv |1\rangle\langle 1|$ (which we can denote at the same time as $P_x \equiv |x\rangle\langle x|$, with $x = 0$ or 1):

$$|x\rangle|e\rangle \longrightarrow P_0 \otimes I\,|x\rangle|e_0\rangle + XP_0 \otimes I\,|x\rangle|e_1\rangle + XP_1 \otimes I\,|x\rangle|e_2\rangle + P_1 \otimes I\,|x\rangle|e_3\rangle \,. \tag{8.3.3}$$

The first two terms are non-zero only when $x = 0$ and the last two are non-zero only when $x = 1$.

At this point, it will help to rewrite the projector P_x in the form

$$P_x = \frac{I + (-1)^x Z}{2} \,. \tag{8.3.4}$$

You can check that this is correct:

$$\begin{aligned} P_0|0\rangle &= \frac{I+Z}{2}|0\rangle = \frac{1}{2}(|0\rangle + |0\rangle) = |0\rangle \\ P_0|1\rangle &= \frac{I+Z}{2}|1\rangle = \frac{1}{2}(|1\rangle - |1\rangle) = 0 \\ P_1|0\rangle &= \frac{I-Z}{2}|0\rangle = \frac{1}{2}(|0\rangle - |0\rangle) = 0 \\ P_1|1\rangle &= \frac{I-Z}{2}|1\rangle = \frac{1}{2}(|1\rangle + |1\rangle) = |1\rangle \,. \end{aligned} \tag{8.3.5}$$

With this version of P_x Eq. (8.3.3) becomes

$$\begin{aligned} |x\rangle|e\rangle \longrightarrow & I \otimes I\,|x\rangle\left(\frac{|e_0\rangle + |e_3\rangle}{2}\right) + Z \otimes I\,|x\rangle\left(\frac{|e_0\rangle - |e_3\rangle}{2}\right) \\ & + X \otimes I\,|x\rangle\left(\frac{|e_2\rangle + |e_1\rangle}{2}\right) + Y \otimes I\,|x\rangle\left(\frac{|e_2\rangle - |e_1\rangle}{2}\right), \end{aligned} \tag{8.3.6}$$

where to get the last term I've used the fact that $XZ = -Y$. We can make (8.3.6) more compact by using $|a\rangle, |b\rangle, |c\rangle$, and $|d\rangle$ to represent the combinations of environmental states in parentheses:

$$|x\rangle|e\rangle \longrightarrow I\otimes I|x\rangle|a\rangle + X\otimes I|x\rangle|b\rangle + Y\otimes I|x\rangle|c\rangle + Z\otimes I|x\rangle|d\rangle\,. \tag{8.3.7}$$

Since this linear transformation holds for both $x = 0$ and $x = 1$, it also holds for any linear combination $|\psi\rangle$ of the two:

$$|\psi\rangle|e\rangle \longrightarrow I\otimes I|\psi\rangle|a\rangle + X\otimes I|\psi\rangle|b\rangle + Y\otimes I|\psi\rangle|c\rangle + Z\otimes I|\psi\rangle|d\rangle\,. \tag{8.3.8}$$

Finally, if you have n qubits interacting with the environment, you can show in a straightforward way (see Mermin) that (8.3.8) generalizes to include terms like

$$(A_{n-1}\otimes A_{n-2}\ldots A_0)\otimes I|x\rangle|\text{env.}\rangle\,,$$

(where the A's are I, X, Y, and Z) and that if the interaction with the environment is weak enough so that at most one of the A's differs from I, you end up with

$$|\Psi\rangle|e\rangle \longrightarrow I\otimes I|\Psi\rangle|a\rangle + \sum_{i=0}^{n-1}\left(X_i\otimes I|\Psi\rangle|b_i\rangle + Y_i\otimes I|\Psi\rangle|c_i\rangle + Z_i\otimes I|\Psi\rangle|d_i\rangle\right). \tag{8.3.9}$$

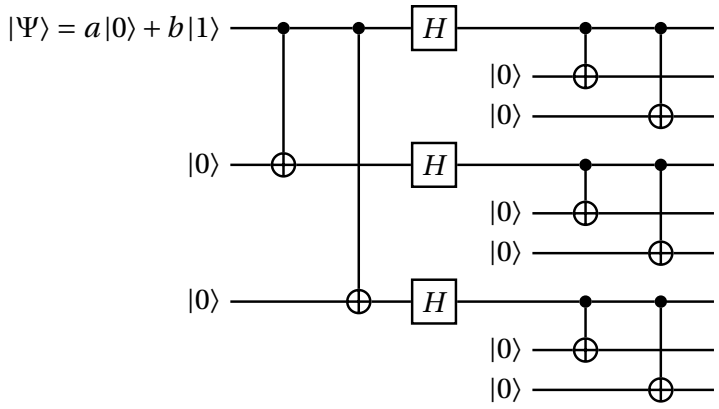
This means that we're restricting ourselves to one-qubit errors, though we could generalize all what we've done so that we don't have to.

What's the point of all this? Well, if we can construct a measurement on the qubits that can distinguish $|\Psi\rangle, X_0|\Psi\rangle, X_1|\Psi\rangle, \ldots, Y_0|\Psi\rangle, Y_1|\Psi\rangle, \ldots, Z_0|\Psi\rangle, Z_1|\Psi\rangle, \ldots$ from one another, we will collapse the right hand side of (8.3.9) into just one of its terms — a single tensor product of a one-qubit state with one of the environmental states — by doing that measurement. Having done it, we can then correct the error. If (8.3.9) contained only the X terms, our bit-flip code would do the trick, and if it contained only the Z terms, our phase-flip code would. But to fully correct all single-qubit errors, we need to be able to correct both kinds, plus Y errors, with a single circuit. We'll see one way to do that next.

Note, by the way, that because the environmental states don't have to be normalized, the form in (8.3.9) includes the small changes we discussed in (8.1.5). It's still the case that a measurement of the some some auxiliary bits) that can distinguish the various kinds of errors (called "syndromes") will collapse the state into one in which the encoded bits have a single kind of error that can be corrected.

8.4 Shor's 9-Qubit Code

So how do we correct arbitrary errors of the form in (8.3.9)? Shor's idea, the first in the field, was to "concatenate" the bit-flip and phase-error codes into a circuit that could correct both. His scheme uses the encoding circuit here:



I will let you check for yourselves that the state that emerges from this circuit is

$$|\bar{\Psi}\rangle = a|\Phi_0\rangle|\Phi_0\rangle|\Phi_0\rangle + b|\Phi_1\rangle|\Phi_1\rangle|\Phi_1\rangle\,, \tag{8.4.1}$$

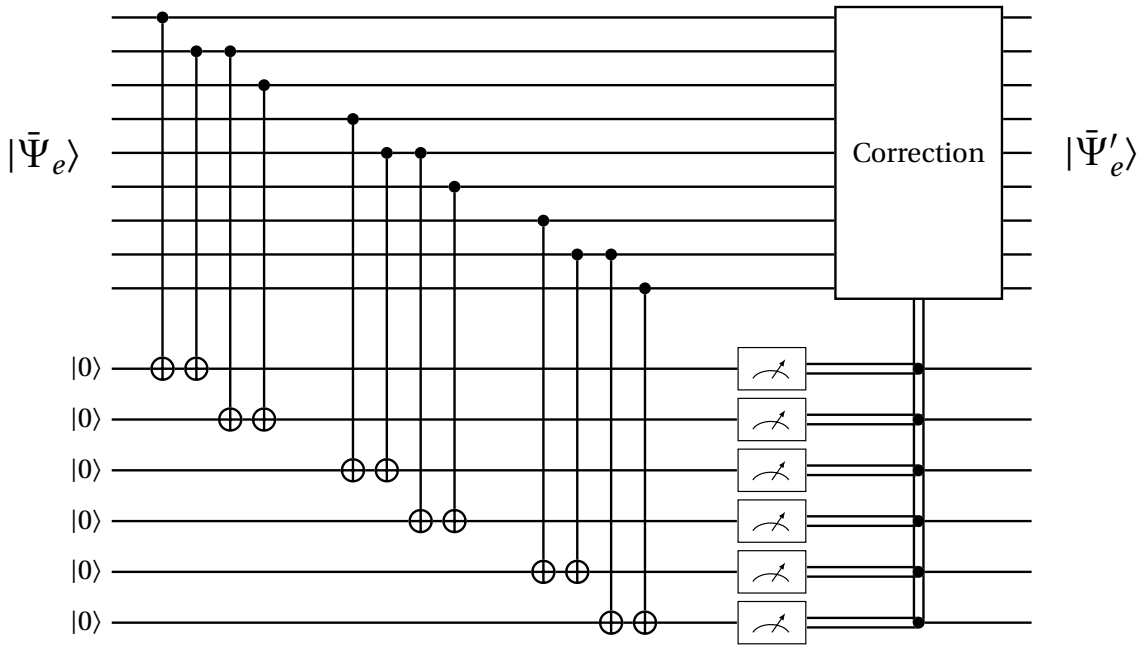
where

$$\begin{aligned} |\Phi_0\rangle &= \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle) \\ |\Phi_1\rangle &= \frac{1}{\sqrt{2}}(|000\rangle - |111\rangle)\,. \end{aligned} \tag{8.4.2}$$

To see this most easily, look at the what happens to the two terms in $|\Psi\rangle$ separately.

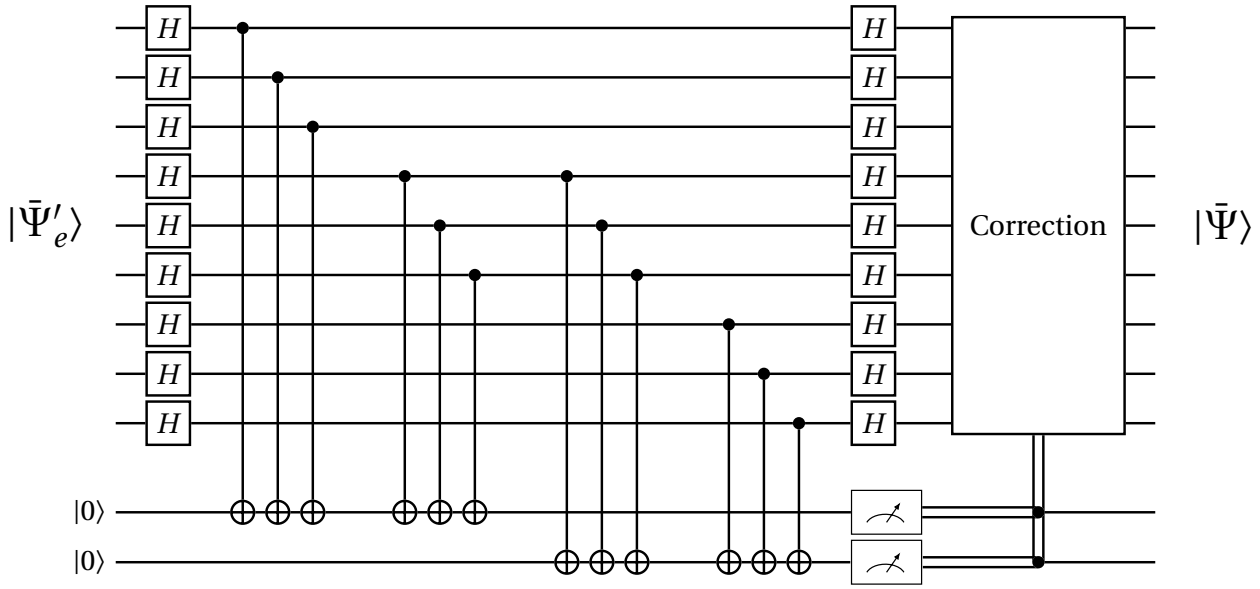
The form of Eqs. (8.4.1) and (8.4.1) are suggestive. It looks like $|\bar{\Psi}\rangle$ contains a superposition of terms containing three blocks each, with each block ($|\Phi_0\rangle$ or $|\Phi_1\rangle$) itself encoded in what sort of resembles a three qubit version of $|+\rangle$ or $|-\rangle$. But these blocks, given in (8.4.2), are not $|+++\rangle$ or $|---\rangle$, so figuring out how to identify the syndrome will not be quite as simple as just stealing from the circuits we've already examined.

Well, actually, diagnosing bit-flip error, *is* just like what we've done before. The circuit below will identify and correct a flip of any of the nine qubits.



Why? Well suppose qubit 4 is flipped. Then we know that the syndrome-diagnosing part of the circuit will do nothing to block 1, because it contains only linear combinations of $|000\rangle$ and $|111\rangle$, meaning that the four NOTS it controls will always leave its auxiliary bits in the state $|00\rangle$. The same is true of block 3. But block 2, as we also know from our earlier analysis, will leave the auxiliary qubits connected to it in the state $|10\rangle$. After measuring the syndromes, we will know the error is in bit 4 and can correct it. And the procedure will work no matter which qubit gets flipped.

What does this circuit do to a phase error, say in qubit 4? Nothing, for the same reason our first three-qubit encoding didn't allow the detection of phase errors. So if we follow the nine-qubit circuit we just looked at with a phase-error detecting one, we'll take care of both kinds of errors. It's the phase detection, though, that is a bit more complicated than with three-qubit encoding. I claim that this circuit will do the job:



To see that this works, consider the "parity" of each of the three three-qubit blocks multiplying a and b. The parity of a string of bits is defined to be $(-1)^{n_1}$, where n_1 is the number of 1's in the string. So a string with an even number of 1's has parity 1 and string with an odd number of 1's has parity -1. The first important thing to note is that $W|\Phi_0\rangle \equiv H\otimes H\otimes H|\Phi_0\rangle$ has parity 1 — all the terms with an odd number of 1's cancel — while $W|\Phi_1\rangle$ has parity -1. You should verify these facts for yourself. The second important thing is that each group of three controlled NOTs does nothing to the auxiliary bit to which it's attached if the block that the group is coming from has parity 1, and flips the auxiliary bit if the block has parity -1. You should also verify this fact. It means that the first auxiliary qubit will be flipped if and only if the parities of the first and second blocks are different. The second auxiliary qubit will be flipped if and only if the parities of the second and third blocks are different. Thus, the parities 1 and -1 of the three blocks play the same role as $|0\rangle$ and $|1\rangle$ of the three qubits in the three-bit encoding we first used to identify bit flips.

Finally, look what a phase error on a qubit does to the parity of the block that the qubit is in. You can check that, because a phase error in any of the three qubits has the

same effect, namely changing the sign in front of the $|111\rangle$,

$$|\Phi_0\rangle \xleftrightarrow{\text{phase error}} |\Phi_1\rangle \,. \tag{8.4.3}$$

Thus, under a phase flip, the nine-qubit state in Eq. (8.4.1) behaves in exactly the same way as the three-qubit state in the second line of Eq. (8.1.2) behaves under a bit flip. That means that our parity-comparing auxiliary bits detect which block has flipped from a $|\Phi_0\rangle$ to a $|\Phi_1\rangle$ (or vice versa) in the same way that our bit-flip-comparing auxiliary bits detect which qubit has flipped from $|0\rangle$ to $|1\rangle$ (or vice versa). So this 9-qubit circuit detects phase errors in the same way our first three-qubit circuit detected bit-flip errors.

What if a qubit undergoes a Y error? I'll leave it to you to verify that, because $Y = -XZ$, the bit-flip-correcting circuit followed by the phase-error-detecting circuit do exactly what is needed. And that's how the nine-qubit encoding scheme works.

Multi-bit errors are more complicated and require more encoding qubits. Error correction on a real quantum computer would demand, instead of the 17 physical qubits per one qubit of information (a "logical qubit") as in the example above (nine for the encoding, six for bit-flip correction and two for phase-error correction), about 1000 physical qubits per logical qubit. But you can read all about that on your own. For us, that's all.