Quantum Information Processing

Course notes for

QIC 710 / CS 768 / PHYS 767 / CO 681 / AMATH 871 / PMATH 871

Richard Cleve

Institute for Quantum Computing & Cheriton School of Computer Science

University of Waterloo

September 3, 2025

Abstract

These notes introduce some of the basics of quantum information processing, with intuition and technical definitions, aimed at anyone with a solid understanding of linear algebra and probability theory. They are in three parts

Part I: A primer for beginners

Part II: Quantum algorithms

Part III: Quantum information theory

where the second and third parts can be read in either order.

I have attempted to make these notes accessible to the diverse community of students who usually take the course. I suggest that readers skim over (or skip) any parts that they already have a good feel for.

© 2025 by Richard Cleve. All rights reserved.

Contents

Part I
A primer for beginners

1 Preface

The goal here is to explain the basics of quantum information processing, with intuition and technical definitions. To be able to follow this, you need to have a solid understanding of linear algebra and probability theory. But no prior background in quantum information or quantum physics is assumed.

You'll see how information processing works on systems consisting of quantum bits (called *qubits*): the kinds of manoeuvres that are possible; and also what is *not* possible. You'll see this in the context of some simple communication scenarios.

Although the scenarios arising in this part are small toy problems, they are part of a foundation. This will help you internalize the more dramatic results that you'll see in *Part II: quantum algorithms* and *Part III: quantum information theory*.

If you are already past the beginner stage

Then you might skip several sections. Perhaps you'll find sections 5 and 12 intriguing. And you should ensure that you understand these topics from *Part I*:

- State distinguishing problems (specifically, sections 4.1 and 4.3)
- Controlled-U and CNOT gates (sections 6.8, 6.9, and 8.5)
- Superdense coding (section 7)
- Projective and local measurements (sections 8.1 and 8.2)
- Isometries (section 9)
- Teleportation (section 10)
- The no-cloning theorem (section 11.2)

2 What is a qubit?

In this section we are going to see how single quantum bits—called qubits—work. Some of you may have already seen that the state of a qubit can be represented as a 2-dimensional vector (or a 2×2 "density matrix"). Since there are a continuum of such possible states, it is natural to ask:

Is a qubit digital or analog?

How much information is there in a qubit?

Please keep these questions in mind, as we work our way from bits to qubits.

2.1 A simple digital model of information

To begin with, please take a moment to consider how to answer the question:

What is a *bit*?

Although a valid answer is that a bit is an element of $\{0, 1\}$, I'd like you to think of a bit in an *operational* way, as a *system* that can *store* an element of $\{0, 1\}$ and from which the information can be *retrieved*. There are also other operations that we might want to be able to perform on a bit, such as modifying the information stored in it in some systematic way.

I happen to own a little 128 gigabyte USB flash drive that looks like this.

Figure 1: My 128 GB USB flash drive.

Think of a *bit* as a flash drive containing *just one single bit of information*. Let the blue box in figure 2 denote such a system.

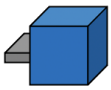

Figure 2: Think of a *bit* as a USB drive containing one single bit of information.

We will imagine a few simple devices that perform operations on such bits. First, imagine a device that enables us to *set* the value of a bit to 0 or 1.

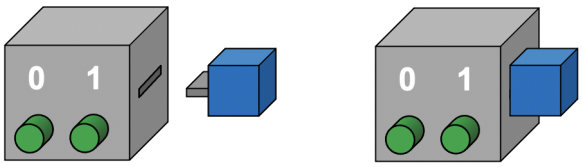

Figure 3: A *set* device enables us to set a bit to 0 or 1.

We plug our bit into that device and then we push one of the two green buttons to set the state to either 0 or 1. Suppose we push the button on the left to set the state to 0.

Later on, we (or someone else) might want to read the information stored in a bit. Imagine a *read* device that enables this.

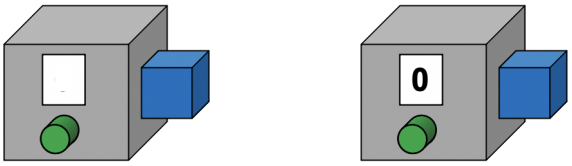

Figure 4: Plug a bit into a *read* device and push the activation button to see it's value.

We can plug the bit into that device and then push the activation button. This causes the bit's value to appear on a screen, so that we can see it.

A third type of device is one that transforms the state of a bit in some way. For example, for a NOT device, we plug the bit in and, when we push the button, the state of the bit flips (0 changes to 1 and 1 changes to 0).

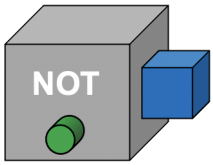

Figure 5: A NOT device enables us to flip the value of a bit.

This transformation is called NOT because it performs a logical negation, where we associate 0 with "false" and 1 with "true". Note that, for this kind of operation, we don't care about seeing what the value of the bit is, as long as that value gets negated.

OK, that's more or less what conventional information processing is like—albeit with many more bits in play and much more complicated operations.

2.2 A simple analog model of information

Next, let's consider an analog information storage system. It has a continuum of possible states (perhaps a voltage that can be anywhere within some range). We can abstractly think of the state of the system as any real number between 0 and 1 (that is, in the interval $[0, 1]$). We'll use a different color to distinguish this from the bit.

Figure 6: An analog USB drive that stores a value in the interval $[0, 1]$.

Let the red box in figure 6 represent such a system, an analog memory.

Imagine a device that *sets* the state of the analog memory. We plug our system

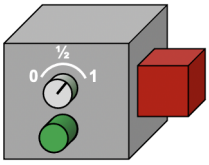

Figure 7: An analog *set* device.

into it. Suppose that there is some kind of dial that can be continuously rotated to specify any number between 0 and 1. Then we press the activation button and the state of the system becomes the value that we selected.

We can also imagine reading the state of such a system. Here the *read* device has

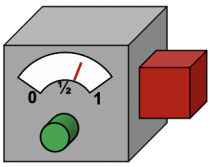

Figure 8: An analog *read* device.

an analog display depicted as a meter. When we press the button the needle goes to a position between 0 and 1, corresponding to the state.

And we can also imagine an analog *transformation* that, when activated, applies

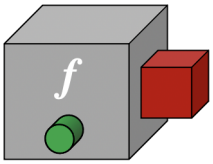

Figure 9: An analog *f-transformation* device.

some function $f : [0, 1] \to [0, 1]$ (for example, mapping x to x^2 or x to $1 - x$).

The real numbers are a mathematical idealization. In any implementation, there will be a certain level of limited precision for all of the operations. But such analog devices can be useful even if their precision isn't perfect. Moreover, in principle, one could make the level of precision very high. The resulting system may be very expensive to manufacture, but it could contain a lot of information.

2.3 A simple probabilistic digital model of information

Before considering quantum bits, let's introduce randomness into our notion of a bit.

Suppose that the state of our bit is the result of some random process, so there's a probability that the system is in state 0 and a probability that it's in state 1. Of course the probabilities are greater than or equal to 0 and they sum to 1. Let's put aside the question of what probabilities really mean. I'm going to assume that you already have some understanding of this.

Now imagine a new kind of device to *randomly set* the value of a bit, where some probability value, between 0 and 1, is selected by rotating a dial (within some precision, of course).

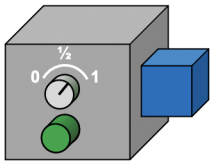

Figure 10: A *probabilistic set* device.

When we activate, the bit gets set to 1 with the probability that we selected; and otherwise it gets set to 0.

Now, from our perspective, if we know how the dial was set, there's a specific probability distribution, with components p_0 and p_1, and the state of the system is best described by this probability vector

$$\begin{bmatrix} p_0 \\ p_1 \end{bmatrix}. \tag{1}$$

But note that the *actual* state is either 0 or 1 (we just don't know which). The probability vector is a useful way for us to think about the state given what we know (and don't know).

Notice that the probabilistic digital model has an analog flavour. There are a continuum of possible probability distributions. The set device for analog (figure 7) and the set device for probabilistic digital (figure 10) are superficially similar: they both have a dial for selecting a value between 0 and 1. However, what the devices actually *do* is very different.

Suppose that, later on, we insert our bit into a read device—which is the same read device as in figure 4. After we press the activation button, the actual value of

the bit appears on the screen. Once we see the value of the bit, we change whatever probability vector we might have associated with it: the component corresponding to what we saw becomes 1 and the other component becomes 0. Let's refer to this change as the "collapse of the probability vector".

Note that, if we activate the read device a second time we will just see the same value we saw the first time—as opposed to another independent sample. To be clear, what the bit contains is the outcome of the original random process for setting the bit. It does not contain information about the random process itself.

Also, if we didn't know what probability values p_0 and p_1 were used when the bit was set then reading the bit does not provide us with those values. After reading the bit, all we can do is make some statistical inferences. For example, if the outcome of the read operation is 1 then we can deduce that p_1 could not have been 0. This is very different from the analog model, where we can actually see the value of the continuously varying parameter using a read device.

There are also transformations, like the NOT operation, and, more generally, any 2×2 stochastic matrix makes sense as a transformation.

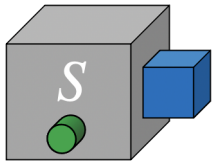

Figure 11: A *stochastic* transformation, where S is some stochastic matrix.

A 2×2 stochastic matrix is of the form

$$S = \begin{bmatrix} s_{00} & s_{01} \\ s_{10} & s_{11} \end{bmatrix}, \tag{2}$$

where $s_{00}, s_{01}, s_{10}, s_{11} \geq 0$, $s_{00} + s_{10} = 1$, $s_{01} + s_{11} = 1$. In other words, each column of S is a valid probability distribution. Applying S changes state 0 to $\left[\begin{smallmatrix} s_{00} \\ s_{10} \end{smallmatrix}\right]$ and state 1 to $\left[\begin{smallmatrix} s_{01} \\ s_{11} \end{smallmatrix}\right]$. If our knowledge of the state is summarized by the probability distribution $\left[\begin{smallmatrix} p_0 \\ p_1 \end{smallmatrix}\right]$ then applying S changes our knowledge to $S\left[\begin{smallmatrix} p_0 \\ p_1 \end{smallmatrix}\right]$.

OK, that's essentially what information processing with bits is like when we allow random operations (again, with many more bits in play and much more complicated operations).

2.4 A simple quantum model of information

So how do *quantum* bits fit in? Are quantum bits like probabilistic bits or are they like analog? In fact, they are neither of these. Quantum information is an entirely different category of information. But it will be worth comparing it to probabilistic digital and analog.

A quantum bit (or *qubit*) has a *probability amplitude* associated with 0 and with 1. Probability amplitudes (called *amplitudes* for short) are different from probabilities. They can be negative—in fact they can be complex numbers. As long as they satisfy the condition that their absolute values squared sum to 1. In other words the amplitude vector, written here with components α_0 and α_1, is a vector

$$\begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} \in \mathbb{C}^2 \tag{3}$$

whose Euclidean[1] length is 1 (also called a *unit* vector).

OK, that's a definition, but it's natural to ask: what do these amplitudes actually *mean*? Our approach to answering this question will be *operational*. What I mean by this is that we'll consider what happens to qubits when basic operations similar to set, read, and transform are performed. We'll develop an understanding of qubits by seeing them in action.

Later on, it will become clear that, unlike with probabilities, the explicit state of a qubit is not 0 or 1; it works better to think of the amplitude vector $\left[\begin{smallmatrix} \alpha_0 \\ \alpha_1 \end{smallmatrix}\right]$ *as the explicit state*. In this one respect, quantum digital states resemble our analog system, where the explicit state is the continuous value.

Now, let's see qubits in action. We have our quantum memory, which we will denote as a purple box, containing a qubit.

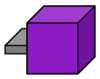

Figure 12: A quantum memory containing a qubit.

To begin with, imagine a device that enables us to *set* the state of a qubit to any amplitude vector.

[1]The Euclidean length of a vector $\left[\begin{smallmatrix} \alpha_0 \\ \alpha_1 \end{smallmatrix}\right]$ is defined as $\sqrt{|\alpha_0|^2 + |\alpha_1|^2}$.

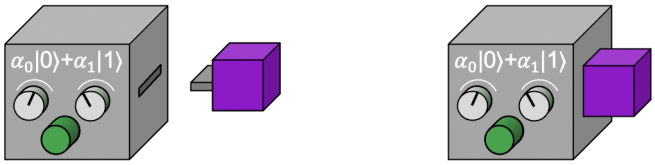

Figure 13: Plug the qubit into a *set* device, set the dials, and then push the activation button to set the state of the qubit.

The device has two dials that we can rotate. Why two? Because there are two real degrees of freedom for all amplitude vectors: the amplitudes α_0 and α_1 (which are complex numbers) can be expressed in a polar form

$$\alpha_0 = \sin(\theta) \tag{4}$$

$$\alpha_1 = e^{i\phi}\cos(\theta) \tag{5}$$

which is in terms of two[2] angles. So we can tune the two dials to specify any state (within some precision), and then we press the activation button and the qubit is set to the state that we specified.

Next, the quantum analogue of the read device is called the *measure* device.

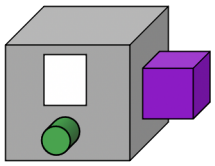

Figure 14: Quantum *measure* device.

We're going to consider this device carefully. Recall that the state of the qubit is described by an amplitude vector $\left[\begin{smallmatrix}\alpha_0\\ \alpha_1\end{smallmatrix}\right]$. What happens during a measurement is:

1. The outcome displayed on the screen is either a 0 or a 1, with respective probabilities the $|\alpha_0|^2$ and $|\alpha_1|^2$. Note that this makes perfect sense as a probability distribution, because these quantities sum to 1.

2. Also, the amplitude vector "collapses" towards the outcome in a manner similar to the way that a probability vector collapses when we read the value of a bit. The amplitude for the outcome becomes 1 and the other amplitude becomes 0.

[2]Perhaps you noticed that there are actually *three* degrees of freedom; however, it turns out that one of them doesn't matter (this will be explained in section 4.2).

This is depicted in Figure 15.

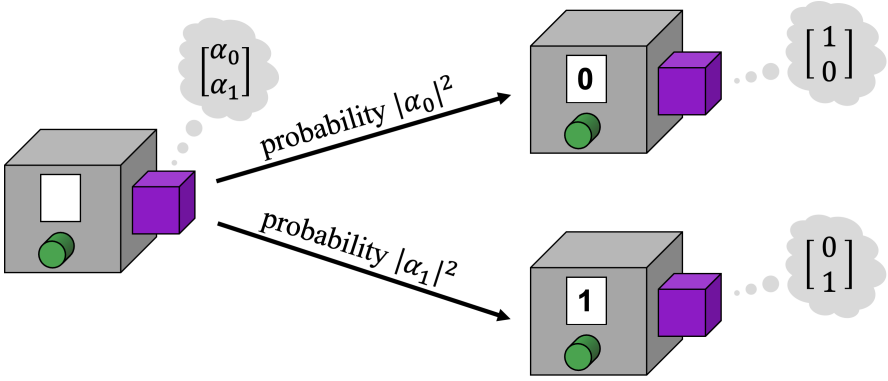

Figure 15: When a measure device is activated, there are two possible outcomes.

For example, suppose we press the button and the outcome is 0 (an outcome that occurs with probability $|\alpha_0|^2$). Then 0 is displayed on the screen and the state of the qubit changes from $\left[\begin{smallmatrix}\alpha_0\\ \alpha_1\end{smallmatrix}\right]$ to $\left[\begin{smallmatrix}1\\ 0\end{smallmatrix}\right]$. The original amplitudes α_0 and α_1 are lost. In this sense, the measurement process is a destructive operation. And there's no point in measuring the qubit a second time; we would just see the exact same result, 0, again.

It should be clear that, if we don't know the state $\left[\begin{smallmatrix}\alpha_0\\ \alpha_1\end{smallmatrix}\right]$ of a qubit, then measuring it does not enable us to extract the amplitudes α_0 and α_1. In this respect, qubits resemble the bits in our probabilistic digital system.

Considering these two operations, set and measure, you might wonder: what's the point of these amplitudes? Amplitudes seem like square roots of probabilities. When we measure, the outcome probabilities depend only on the absolute values of the amplitudes squared. So what relevant information is there in those square roots? In fact, if we stopped with these two operations, set and measure, then qubits would be essentially the same as probabilistic bits.

But qubits are interesting because we can also perform transformations like rotations on amplitude vectors, which essentially change the coordinate system in which subsequent measurements are made. Note that, if you rotate a vector of length 1, it's still a vector of length 1, so the validity of quantum states is preserved. In fact, the allowable transformations are *unitary operations*, which are kind of like "generalized rotations," and include operations like reflections too. If U is a specific 2×2 unitary matrix then Figure 16 shows how we denote the device that performs the operation "multiply the state vector by U."

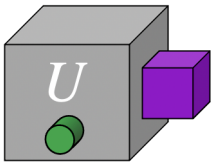

Figure 16: A *unitary* operation, where U is a 2×2 unitary matrix, transforms a state by U.

We'll shortly see (in section 3.3) a formal definition of *unitary* and some interesting manoeuvres involving unitary operations in subsequent sections.

Together, these three kinds of operations—set, measure, and unitary—are essentially the building blocks of quantum information processing. We'll see that all the strange and interesting feats that can be performed in quantum information and quantum computing are based on these operations—and similar ones involving more qubits.

Finally, a comment about terminology. What I've been calling "probabilistic" is commonly known as "classical." The word "classical" is a reference to classical physics, the physics that existed before the advent of quantum physics. So we have *classical information* and *classical bits* vs. *quantum information* and *qubits*.

3 Notation and terminology

We now have a basic picture of how qubits work, but there are a few details to fill in, and we'll spend a little time with that.

3.1 Notation for qubits (and higher dimensional analogues)

First, let's briefly go over some notation and further terminology. Recall that the state of a qubit is its amplitude vector, a unit vector $\left[\begin{smallmatrix}\alpha_0\\ \alpha_1\end{smallmatrix}\right] \in \mathbb{C}^2$. This state is commonly denoted using the *bra-ket* notation as $\alpha_0 \left|0\right\rangle + \alpha_1 \left|1\right\rangle$ (it's also called the *Dirac notation*, after Paul Dirac). The strange looking parentheses (with the angle bracket on the right side) are called *kets*, and $\left|0\right\rangle$ and $\left|1\right\rangle$ are shorthand for the basis vectors, which are orthonormal (where *orthonormal* means orthogonal and of unit length). Figure 17 illustrates the geometric arrangement of the vectors $\left|0\right\rangle$, $\left|1\right\rangle$, and $\alpha_0 \left|0\right\rangle + \alpha_1 \left|1\right\rangle = \left[\begin{smallmatrix}\alpha_0\\ \alpha_1\end{smallmatrix}\right]$ for a generic quantum state vector.

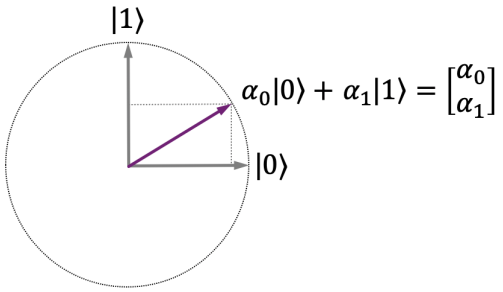

Figure 17: Geometric view of the computational basis states $\left|0\right\rangle$, $\left|1\right\rangle$, and a superposition $\left[\begin{smallmatrix}\alpha_0\\ \alpha_1\end{smallmatrix}\right]$.

Note that figure 17 is a schematic because $\left[\begin{smallmatrix}\alpha_0\\ \alpha_1\end{smallmatrix}\right] \in \mathbb{C}^2$, rather than $\mathbb{R}^2$. The basis vectors $\left|0\right\rangle$ and $\left|1\right\rangle$ are commonly referred to as the *computational basis states*. For quantum states, the linear combinations $\alpha_0 \left|0\right\rangle + \alpha_1 \left|1\right\rangle$ are also called *superpositions*.

More generally, in higher-dimensional systems (which will come up soon), any symbol within a ket denotes a column vector of unit length, like

$$\left|\psi\right\rangle = \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{d-1} \end{bmatrix}, \tag{6}$$

where $\| \left|\psi\right\rangle \| = \sqrt{|\alpha_0|^2 + |\alpha_1|^2 + \cdots + |\alpha_{d-1}|^2} = 1$.

A *bra* is like a ket, but written with the angle bracket on the left side, and it denotes the conjugate transpose of the ket with the same label. Taking the conjugate transpose of a column vector yields a row vector whose entries are the complex conjugates of the original entries, like

$$\langle\psi| = \begin{bmatrix} \bar{\alpha}_0 & \bar{\alpha}_1 & \cdots & \bar{\alpha}_{d-1} \end{bmatrix}. \tag{7}$$

A bra is always a row vector of unit length.

The *inner product* of a two kets is written as a bra-ket, or *bracket*, which can be viewed as shorthand for the product of a row matrix with a column matrix. If

$$|\phi\rangle = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{d-1} \end{bmatrix} \tag{8}$$

then the inner product of $|\psi\rangle$ and $|\phi\rangle$ is the bracket

$$\langle\psi|\phi\rangle = \langle\psi| \cdot |\phi\rangle = \begin{bmatrix} \bar{\alpha}_0 & \bar{\alpha}_1 & \cdots & \bar{\alpha}_{d-1} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{d-1} \end{bmatrix} \tag{9}$$

$$= \sum_{k=0}^{d-1} \bar{\alpha}_k \beta_k. \tag{10}$$

Recall that, for inner products of complex-valued vectors, one takes the complex conjugates of the entries of one of the vectors.

3.2 Normalization convention for quantum state vectors

Sometimes we denote states like $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ as simply $|0\rangle + |1\rangle$ to reduce clutter. More generally, we can write states as (non-zero) vectors that are not normalized, where we use the *normalization convention* that unnormalized state vectors are understood be divided by their norm. An unnormalized $|\psi\rangle$ is interpreted as the quantum state $|\psi\rangle / \| |\psi\rangle \|$.

3.3 A closer look at unitary operations

Let U be a square matrix. Here are three equivalent definitions of unitary.

The first definition is in terms of a useful geometric property: U is unitary if it preserves angles between unit vectors. For any two states, there is an angle between them, which is determined by their inner product, and the property is expressed in terms of inner products.

Definition 3.1. A square matrix U is *unitary* if it preserves inner products. That is, for all $|\psi_1\rangle$ and $|\psi_2\rangle$, the inner product between $U|\psi_1\rangle$ and $U|\psi_2\rangle$ is the same as the inner product between $|\psi_1\rangle$ and $|\psi_2\rangle$.

The second definition makes it easy to recognize unitary matrices.

Definition 3.2. A square matrix U is *unitary* if its columns are orthonormal (which is equivalent to its rows being orthonormal).

Some well-known examples of 2×2 unitary matrices are: the *rotation* by angle θ

$$R_\theta = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{11}$$

and the *Hadamard* transform

$$H = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}, \tag{12}$$

which is not a rotation (but H is a *reflection*). Three further examples are the *Pauli* matrices[3]

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \quad \text{and} \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}. \tag{13}$$

The Pauli X is sometimes referred to as a *bit flip* (or NOT operation), since $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$. Also, Z is sometimes referred to as a *phase flip*.

The third definition of unitary, is useful in calculations and is commonly seen in the literature.

Definition 3.3. A square matrix U is *unitary* if $U^*U = I$, where U^* is the conjugate transpose[4] of U (the transpose of U with all the entries conjugated).

[3] An older notation for the Pauli matrices, commonly used in physics, is σ_X, σ_Y, and σ_Z.

[4] An alternative notation for U^*, commonly used in physics, is $U^\dagger$.

It remains to show that the above three definitions of unitary are equivalent:

Exercise 3.1 (fairly straightforward)**.** Show that the above three definitions of unitary are indeed equivalent.

3.4 A closer look at measurements

Now, let's look at measurements again. Let our qubit be in some state $\alpha_0 |0\rangle + \alpha_1 |1\rangle = \left[\begin{smallmatrix}\alpha_0\\ \alpha_1\end{smallmatrix}\right]$ (where $|\alpha_0|^2 + |\alpha_1|^2 = 1$). Then the result of the measurement is the following:

- With probability $|\alpha_0|^2$, the outcome is 0 and the state collapses to $|0\rangle$.
- With probability $|\alpha_1|^2$, the outcome is 1 and the state collapses to $|1\rangle$.

Let's look at this geometrically, in figure 18.

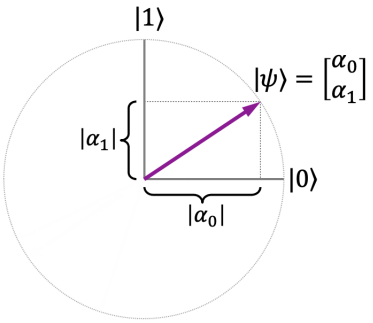

Figure 18: The outcome probabilities of a measurement depend on the projection lengths squared on the computational basis states.

We have a 2-dimensional space with computational basis $|0\rangle$ and $|1\rangle$. An arbitrary state has a *projection* on each basis state. What happens in a measurement is that the state collapses to each basis state with probability equal to the projection-length squared.

The geometric perspective suggests some potential variations in our definition of a measurement. For example, there's no fundamental reason why the computational basis states should have special status. We can imagine basing a measurement on some other orthonormal basis, different from the computational basis. For example, consider the orthonormal basis $|\phi_0\rangle$ and $|\phi_1\rangle$ in figure 19.

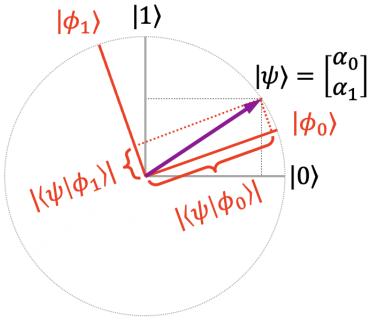

Figure 19: Measurement with respect to an alternative orthonormal basis, $|\phi_0\rangle$ and $|\phi_1\rangle$.

Any state has a projection on each basis vector and, although the projection lengths squared are different for this basis, they still add up to 1. We can *define* a new measurement operation that projects the state being measured $|\psi\rangle$ to each of these basis vectors with probability the projection lengths squared:

- With probability $|\langle\psi|\phi_0\rangle|^2$, the outcome is 0 and the state collapses to $|\phi_0\rangle$.
- With probability $|\langle\psi|\phi_1\rangle|^2$, the outcome is 1 and the state collapses to $|\phi_1\rangle$.

One way of thinking about what unitary operations do is that they permit us to perform measurements with respect to any alternative orthonormal basis. We have our basic measurement operation (which is with respect to the computational basis). If we want to perform a measurement with respect to a different orthonormal basis $|\phi_0\rangle = U|0\rangle$ and $|\phi_1\rangle = U|1\rangle$ then we carry out the following procedure:

1. Apply U^* to map the alternative basis to the computational basis ($|0\rangle$ and $|1\rangle$).
2. Perform a basic measurement (with respect to the computational basis).
3. Apply U to appropriately adjust the collapsed state (to one of $|\phi_0\rangle$ and $|\phi_1\rangle$).

So that's a nice way of seeing the role of unitary operations: they change the coordinate system, thereby releasing us from being tied to measuring in the computational basis.

A final comment here is that there are more exotic measurements than this, where the state is first embedded into a larger-dimensional space. Then a unitary operation and measurement are made in that larger space. We'll be seeing these types of measurements later on (in section 9.2).

4 Introduction to state distinguishing problems

Define the *plus* state and *minus* state as

$$|+\rangle = \tfrac{1}{\sqrt{2}}|0\rangle + \tfrac{1}{\sqrt{2}}|1\rangle \tag{14}$$

$$|-\rangle = \tfrac{1}{\sqrt{2}}|0\rangle - \tfrac{1}{\sqrt{2}}|1\rangle\,. \tag{15}$$

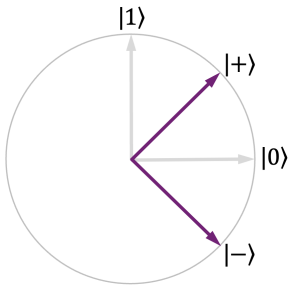

Figure 20: Geometric depiction of the states $|+\rangle$, and $|-\rangle$.

What happens if a qubit in one of these two states is measured with respect to the computational basis? Since $(\frac{1}{\sqrt{2}})^2 = (-\frac{1}{\sqrt{2}})^2 = \frac{1}{2}$, the result of the measurement is a uniformly distributed random bit, for both states.

Now, suppose that we're given a qubit whose state is *promised* to be either $|+\rangle$ or $|-\rangle$, but we're not told which one. Is there a process for determining which one it is?

The first observation is that measuring with respect to the computational basis is useless. For either state, the result will be a uniformly distributed random bit. There's no distinction. But, since we can perform unitary operations, we are not shackled to the computational basis. We can apply a rotation by angle 45 degrees, which maps $|+\rangle$ to $|1\rangle$ and $|-\rangle$ to $|0\rangle$. *Then* we measure in the computational basis, which enables us to *perfectly* distinguish between the two cases. We can describe this procedure as *measuring with respect to the orthonormal basis* $|+\rangle$, $|-\rangle$.

4.1 Distinguishing between $|0\rangle$ and $|+\rangle$

Here's another, more subtle, state distinguishing problem. Suppose that we are given a qubit that is either in the $|0\rangle$ state or the $|+\rangle$ state. Can we distinguish between these two cases? The problem is that the states are not orthogonal—so there's no unitary that takes one of them to $|0\rangle$ and the other to $|1\rangle$. In fact, there is no perfect distinguishing procedure for $|0\rangle$ and $|+\rangle$.

Even though we cannot perfectly distinguish between $|0\rangle$ and $|+\rangle$, we can try to maximize the success probability.

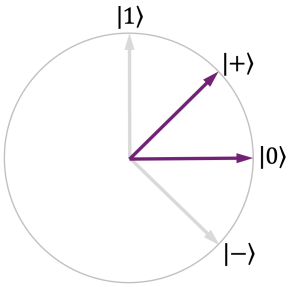

Figure 21: The states $|0\rangle$ and $|+\rangle$.

First note that success probability $\frac{1}{2}$ is trivially attainable by outputting a random guess (without even measuring the state). Success probability $\frac{1}{2}$ is the baseline that we can try to improve on via a measurement.

What happens if we measure in the computational basis? Then it's sensible to guess "0" if the outcome is 0 and guess "+" if the outcome is 1. The success probability of this strategy (which depends on the instance) is

$$\begin{cases} 1 & \text{if the state is } |0\rangle \\ \frac{1}{2} & \text{if the state is } |+\rangle. \end{cases} \tag{16}$$

How good is this? We will consider two different definitions: average-case success probability and worst-case success probability:

Average-case success probability is the average of the success probabilities with respect to some prior probability distribution on the instances. For example, if we assume that $|0\rangle$ and $|+\rangle$ each arise with probability $\frac{1}{2}$ then the average-case success probability of the above measurement procedure is $(\frac{1}{2})(1) + (\frac{1}{2})(\frac{1}{2}) = \frac{3}{4}$.

Worst-case success probability is the minimum of the success probabilities of the instances. The worst-case success probability of the above procedure is $\frac{1}{2}$.

There is a complementary procedure to the procedure of measuring in the computational basis: measure with respect to the $|+\rangle$, $|-\rangle$ basis. Then it's sensible to guess "+" if the outcome is $|+\rangle$ and "0" if the outcome is $|-\rangle$. The success probabilities of the instances are

$$\begin{cases} \frac{1}{2} & \text{if the state is } |0\rangle \\ 1 & \text{if the state is } |+\rangle. \end{cases} \tag{17}$$

The average-case success probability is $\frac{3}{4}$ and the worst-case success probability is $\frac{1}{2}$.

There is a way of combining the two aforementioned measurement procedures to attain *worst-case* success probability $\frac{3}{4}$. The procedure is to randomly choose between the two measurement strategies. In other words, with probability $\frac{1}{2}$, do the above procedure based on measuring with respect to the computational basis, and with probability $\frac{1}{2}$ do the above procedure based on measuring with respect to the $|+\rangle$, $|-\rangle$ basis. The resulting success probabilities for the instances are

$$\begin{cases} \frac{3}{4} & \text{if the state is } |0\rangle \\ \frac{3}{4} & \text{if the state is } |+\rangle. \end{cases} \tag{18}$$

In fact, there is a better strategy than all of the above strategies, whose worst-case success probability is $\cos^2(\pi/8) = 0.853...$. The idea is to measure with respect to the orthonormal basis

$$|\psi_0\rangle = \cos(\pi/8)\,|0\rangle - \sin(\pi/8)\,|1\rangle \tag{19}$$

$$|\psi_1\rangle = \sin(\pi/8)\,|0\rangle + \cos(\pi/8)\,|1\rangle\,. \tag{20}$$

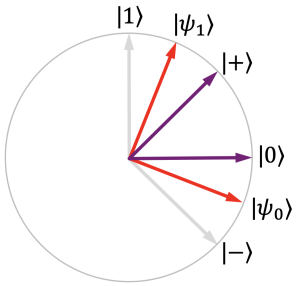

Figure 22: Distinguishing between $|0\rangle$ and $|+\rangle$ by measuring with respect to $|\psi_0\rangle$ and $|\psi_1\rangle$.

Since $|\psi_0\rangle$ and $|\psi_1\rangle$ symmetrically flank $|0\rangle$ and $|+\rangle$, with an angle of 15° ($\pi/8$ radians) at each side, we have $|\langle\psi_0|0\rangle|^2 = |\langle\psi_1|+\rangle|^2 = \cos^2(\pi/8)$. It turns out that $\cos^2(\pi/8)$ is optimal for this state distinguishing problem, though we don't prove it now.[5]

Some simple variants of the problem

Consider the state distinguishing problem where the two states are $|+\rangle$ and $|1\rangle$. It's easy to see that this reduces to the case of $|0\rangle$ and $|+\rangle$ (that we've already solved) by simply applying the unitary operation that rotates clockwise by 45 degrees. More generally, for any two states, $|\phi_0\rangle$ and $|\phi_1\rangle$ for which $\langle\phi_0|\phi_1\rangle = \frac{1}{\sqrt{2}}$, there exists a

[5]We will learn how to prove optimality in section 33.5 of *Part III: quantum information theory.*

unitary operation that maps $|\phi_0\rangle$ to $|0\rangle$ and $|\phi_1\rangle$ to $|+\rangle$, thereby reducing the problem to that of distinguishing between $|0\rangle$ and $|+\rangle$.

What about $|1\rangle$ vs. $|-\rangle$? This case looks a little different, because $\langle 1|-\rangle = -\frac{1}{\sqrt{2}}$ and the angle is not 45 degrees. But, since the *absolute value* of the inner product is $\frac{1}{\sqrt{2}}$, this case also reduces to the case of $|0\rangle$ and $|+\rangle$. This will be explained in the next subsection.

4.2 Global phases

Consider the states $|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and $-|+\rangle = -\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$. Can we distinguish between them? In fact, we cannot. If we apply any unitary operation U to the first state and the result is $\alpha_0|0\rangle + \alpha_1|1\rangle$ then applying U to the second state produces $-\alpha_0|0\rangle - \alpha_1|1\rangle$. If we then measure, the outcome probabilities will be exactly the same in both cases: $|\alpha_0|^2$ and $|\alpha_1|^2$. Since there's no way of distinguishing between the states, we regard them as *the same state*.

The minus sign in $-|+\rangle$ is called a *global phase* ("global" because it's applied to all of the terms of the superposition). A global phase can be any complex number of the form $e^{i\theta}$ for $\theta \in [0, 2\pi]$. We define this equivalence relation on state vectors.

Definition 4.1. Two state vectors $|\psi\rangle$ and $|\phi\rangle$ are deemed *equivalent* if $|\psi\rangle = e^{i\theta}|\phi\rangle$ for some $\theta \in [0, 2\pi]$.

Here's an exercise, if you'd like to get used to this concept.

Exercise 4.1. Partition the following into sets of equivalent states:

$$-\tfrac{1}{\sqrt{2}}|0\rangle + \tfrac{1}{\sqrt{2}}|1\rangle \qquad \tfrac{1}{\sqrt{2}}|0\rangle - \tfrac{1}{\sqrt{2}}|1\rangle \qquad \tfrac{1}{\sqrt{2}}|0\rangle + \tfrac{i}{\sqrt{2}}|1\rangle$$
$$\tfrac{i}{\sqrt{2}}|0\rangle + \tfrac{1}{\sqrt{2}}|1\rangle \qquad -\tfrac{1}{\sqrt{2}}|0\rangle + \tfrac{i}{\sqrt{2}}|1\rangle \qquad \tfrac{1}{\sqrt{2}}|0\rangle - \tfrac{i}{\sqrt{2}}|1\rangle$$

Distinguishing between $|1\rangle$ and $|-\rangle$

We are now equipped to return to the problem that was left hanging at the end of section 4.1: that of distinguishing between $|1\rangle$ and $|-\rangle$. Since global phases make no difference in measurement outcomes, this is equivalent to distinguishing between $-|1\rangle$ and $|-\rangle$, and the inner product between these states is $\frac{1}{\sqrt{2}}$. So this problem reduces to the problem of distinguishing between $|0\rangle$ and $|+\rangle$ (by rotating counterclockwise by 90 degrees), the problem that was solved in section 4.1.

4.3 Other state distinguishing problems

What follows are some other state distinguishing problems (also called *state discrimination* problems), given as exercises. Since, at this point, we don't have methodologies for proving optimality, you might not know for sure whether the measurement procedures you devise are optimal.

If you're unsure about how to proceed for a pair of states, it might help to start by calculating the absolute value of the inner product between the states. (In exercise 4.4, where there are more than two states, it's more complicated.)

Exercise 4.2. For each of the following pairs of states, devise a measurement procedure with as large a worst-case success probability as you can attain:

1. $|0\rangle$ vs. $\frac{\sqrt{3}}{2}|0\rangle + \frac{1}{2}|1\rangle$.
2. $\cos(\pi/8)|0\rangle + \sin(\pi/8)|1\rangle$ vs. $\sin(\pi/8)|0\rangle + \cos(\pi/8)|1\rangle$.
3. $\cos(\theta)|0\rangle + \sin(\theta)|1\rangle$ vs. $\cos(\theta)|0\rangle - \sin(\theta)|1\rangle$, where $\theta \in [0, \frac{\pi}{2}]$ is a known angle. The procedure should be a function of θ.

The examples of states that have arisen so far have amplitudes that are real numbers; however amplitudes can be complex numbers. Here are some state distinguishing problems with complex amplitudes (where $i = \sqrt{-1} = e^{i\pi/2}$).

Exercise 4.3. For each of the following pairs of states, devise a measurement procedure with as large a worst-case success probability as you can attain:

1. $|0\rangle$ vs. $\frac{i}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$
2. $\frac{i}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ vs. $\frac{1}{\sqrt{2}}|0\rangle + \frac{i}{\sqrt{2}}|1\rangle$
3. $\frac{1}{\sqrt{2}}|0\rangle + \frac{1+i}{2}|1\rangle$ vs. $\frac{1}{\sqrt{2}}|0\rangle + \frac{1-i}{2}|1\rangle$

State distinguishing problems involving more than two states also make sense. Here is an example.

Exercise 4.4. Devise a measurement procedure with as large a worst-case success probability as you can attain for distinguishing between the four states

$$|0\rangle \quad \text{vs.} \quad |1\rangle \quad \text{vs.} \quad |+\rangle \quad \text{vs.} \quad |-\rangle\,.$$

Since there are four states, the baseline success probability that arises from random guessing is $\frac{1}{4}$. Your procedure should do better than that.

In general, state distinguishing problems involving more than two states can be tricky. An interesting state distinguishing problem involving three states arises in the next section (figure 28).

5 Can a qubit store more information than a bit?

Remember one of the questions posed at the beginning of section 2: How much information is there in a qubit? Since a qubit can be in a continuum of explicit states, the amount of information needed to *specify* a quantum state is huge—or even infinite if the precision is perfect. But the measurement operation is very severe, yielding only a discrete outcome like 0 or 1, so we cannot "read out" the continuous value.

A famous theorem in quantum information theory, called Holevo's Theorem[6] (dating back to 1973!) implies that classical information cannot be compressed by encoding it as a quantum state and then recovering the data by a measurement. A more recent related result, due to Nayak,[7] that is simpler to state and to prove, also implies this. These results are technically in an *average-case* setting, where it is assumed that there is a prior probability distribution on the classical data.

Here, I will show that, in a *worst-case* setting, a qubit can slightly outperform a bit for a communication problem. Although the advantage is modest, this is an example of an information processing task where a single qubit can achieve something that a single classical bit cannot.

5.1 On communicating a *trit* using a qubit

A qubit can obviously store a bit (representing 0 as $|0\rangle$ and 1 as $|1\rangle$), but suppose we want to use it to store more information than one bit. The smallest upgrade we could ask for is to store a *trit*, which is an element of $\{0, 1, 2\}$. Can a qubit store a trit? Although a qubit cannot *perfectly* store a trit (intuitively, because there do not exist three mutually orthogonal vectors in two dimensions), I will show you that there is a sense in which a qubit can store *more partial information* about a trit than a classical bit can.

To make the scenario clear, suppose there are two parties, A and B, that we'll personify and refer to as Alice and Bob. Alice receives a trit $a \in \{0, 1, 2\}$ as input and the goal is to convey this information over to Bob. We consider two types of strategies that Alice and Bob can carry out: bit strategies and qubit strategies. We will see that there exists a qubit strategy that outperforms every bit strategy.

[6] A. S. Holevo (1973). "Bounds for the quantity of information transmitted by a quantum communication channel." *Problems of Information Transmission*, **9**(3): 177–183.

[7] A. Nayak (1999). "Optimal lower bounds for quantum automata and random access codes." *Proc. 40th Ann. Symp. on Foundations of Computer Science (FOCS)*, pp. 369–376. arXiv:9904093

In a *bit strategy*, Alice is allowed to send just one *bit* to Bob, from which he should somehow produce a good guess of the value of the trit a.

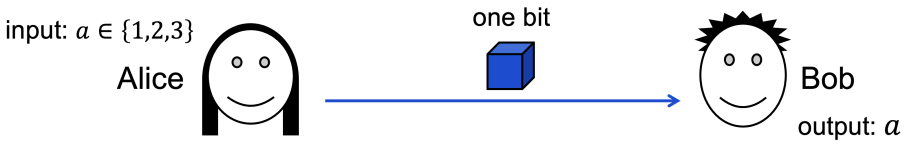

Figure 23: Alice conveying information about a trit to Bob by a *bit strategy*.

In a *qubit strategy*, Alice is allowed to send just one *qubit* to Bob, from which he should somehow produce a good guess of the value of the trit a.

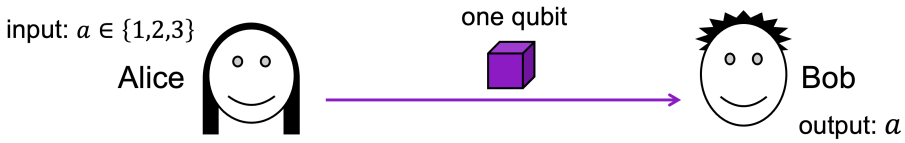

Figure 24: Alice conveying information about a trit to Bob by a *qubit strategy*.

5.1.1 Worst-case success probability of bit strategies

Here is a simple example of a bit strategy. Alice receives her trit and she encodes 0

Figure 25: A simple bit strategy for Alice conveying a trit to Bob.

as 0, 1 as 1, and 2 also as 1. Then Bob decodes to 0 to 0 and 1 to 1. This obviously succeeds for inputs 0 and 1, but fails miserably for input 2.

In the case where the input is a uniformly distributed trit, this strategy succeeds with probability $\frac{2}{3}$. The aforementioned result of Nayak[7] implies that, for uniformly distributed trits, this is optimal among bit strategies as well as qubit strategies.

The situation is more interesting when we consider the *worst-case* input. For any given strategy, this is defined as the smallest success probability for all inputs (instead of the average of the success probabilities). This framework makes sense if Alice and Bob have no idea what distribution Alice's input trit will arise from. Whatever strategy they come up with, the trit *could* be the case where their strategy performs the worst.

Consider the classical bit strategy that we saw in figure 25, whose average-case success probability is $\frac{2}{3}$. What is its worst-case success probability? For the worst-case instance, the success probability is zero! If the trit is 2 then Bob produces the wrong value for sure!

But the worst-case success probability can be improved to $\frac{1}{2}$ as follows.

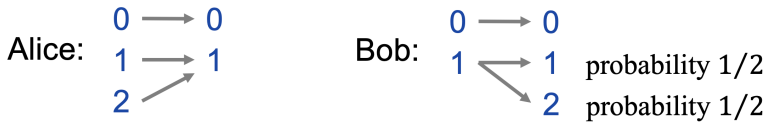

Figure 26: Another classical bit strategy for Alice conveying a trit to Bob.

Bob decodes a 1 *randomly* to either 1 or 2. This bit strategy has worst-case success probability $\frac{1}{2}$. Success probability $\frac{1}{2}$ may seem weak. But if there were no communication from Alice to Bob then the best success probability would be $\frac{1}{3}$. So the bit strategy achieves something: it increases the success probability from $\frac{1}{3}$ to $\frac{1}{2}$.

In fact, success probability $\frac{1}{2}$ is the best possible for bit strategies where Alice and Bob use *local randomness*. Local randomness means that Alice and Bob have separate sources of randomness that are stochastically independent; their randomness is uncorrelated. Alice can use *her* randomness to probabilistically map her trit to a bit, and then, when Bob receives the bit at his end, he can use *his* randomness to probabilistically generate a trit from it. Here is a proof of optimality of $\frac{1}{2}$.

Theorem 5.1. *For any bit strategy where Alice and Bob use local randomness, the best possible worst-case success probability is* $\frac{1}{2}$.

Proof (sketch). Any classical bit strategy for conveying a trit using local randomness is of the following form, where the labeled outgoing edges represent probabilistic

Figure 27: General form of classical bit strategy with local randomness conveying a trit.

choices. Thus, the labels of outgoing edges from each state are probability distributions: for each $a \in \{0,1,2\}$, it holds that $p_{a,0},\ p_{a,1} \geq 0$ and $p_{a,0} + p_{a,1} = 1$; also, for each $b \in \{0,1\}$, it holds that $q_{b,0},\ q_{b,1},\ q_{b,2} \geq 0$ and $q_{b,0} + q_{b,1} + q_{b,2} = 1$. What is

the best possible worst-case success probability attainable by filling in the probability values of such a strategy?

To simplify the analysis, suppose that Alice receives $a \in \{0, 1, 2\}$ as her input trit. From Alice's perspective, there are two relevant probabilities associated with this a: the probability that Bob correctly outputs a if Alice sends bit 0, which is $q_{0,a}$; and the probability that Bob correctly outputs a if Alice sends bit 1, which is $q_{1,a}$. It is clearly optimal for Alice to send the bit b for which $q_{b,a}$ is maximum (and Alice might as well send bit 0 in the case of a tie). Therefore, *even though Alice is allowed to behave probabilistically*, there is an optimal strategy where Alice behaves deterministically.

Now, all Alice's possible deterministic strategies essentially look like the left side of figure 26, where Alice's encoding has a *collision*: two trit values that encode to the same bit value. For any of these deterministic encoding strategies for Alice, whatever Bob's probabilistic decoding procedure is, it will fail with probability at least $\frac{1}{2}$ for one of the trit values where the collision occurs. ■

5.1.2 Worst-case success probability of qubit strategies

What is interesting is that there exists a qubit strategy that succeeds with probability higher than $\frac{1}{2}$ in the worst case. In the qubit strategy, Alice encodes the trit as one of the so-called *trine* states. These are three amplitude vectors in two dimensions with an equal angle of 120 degrees ($\frac{2\pi}{3}$ radians) between them:

$$|\phi_0\rangle = |0\rangle \tag{21}$$

$$|\phi_1\rangle = -\tfrac{1}{2}|0\rangle + \tfrac{\sqrt{3}}{2}|1\rangle \tag{22}$$

$$|\phi_2\rangle = -\tfrac{1}{2}|0\rangle - \tfrac{\sqrt{3}}{2}|1\rangle\,. \tag{23}$$

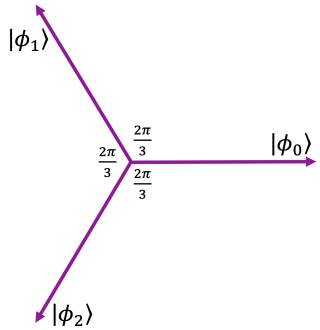

Figure 28: The three trine states.

Theorem 5.2. *There is a qubit strategy that attains worst-case success probability* $\frac{7}{12}$. *To be clear about the model, this strategy employs just a single qubit, that Alice sends to Bob, and the parties may use localized randomness.*

Proof. The qubit strategy is based on Alice encoding the trit $k \in \{0,1,2\}$ as the trine state $|\phi_k\rangle$ and then Bob decoding via the following measurement procedure. Bob randomly chooses an $\ell \in \{0,1,2\}$ and then measures with respect to the orthonormal basis $|\phi_\ell\rangle$, $|\phi_\ell^\perp\rangle$, where

$$|\phi_0^\perp\rangle = |1\rangle \tag{24}$$

$$|\phi_1^\perp\rangle = -\tfrac{\sqrt{3}}{2}|0\rangle - \tfrac{1}{2}|1\rangle \tag{25}$$

$$|\phi_2^\perp\rangle = \tfrac{\sqrt{3}}{2}|0\rangle - \tfrac{1}{2}|1\rangle\,. \tag{26}$$

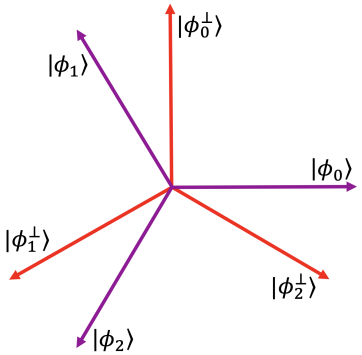

Figure 29: The states $|\phi_0^\perp\rangle$, $|\phi_1^\perp\rangle$, and $|\phi_2^\perp\rangle$ (shown in red).

The outcome of the measurement is either $|\phi_\ell\rangle$ or $|\phi_\ell^\perp\rangle$. If the outcome is $|\phi_\ell\rangle$ then Bob outputs ℓ as his guess for k; if the outcome is $|\phi_\ell^\perp\rangle$ then Bob randomly chooses between the two elements of $\{0,1,2\}$ that are not ℓ as his guess.

How well does this perform? Whatever the trit k is, with probability $\frac{1}{3}$, Bob is lucky and $\ell = k$, in which case Bob is measuring $|\phi_k\rangle$ with respect to the orthonormal basis $|\phi_k\rangle$, $|\phi_k^\perp\rangle$, so his guess will correctly be k for sure. With probability $\frac{2}{3}$, Bob is not so lucky and $\ell \neq k$. Conditional on that case arising, the measurement outcome is $|\phi_\ell^\perp\rangle$ with probability $\frac{3}{4}$ and, conditional on that, Bob guesses k with probability $\frac{1}{2}$. Considering all the conditional probabilities, the success probability is

$$\left(\tfrac{1}{3}\right)\left(1\right) + \left(\tfrac{2}{3}\right)\left(\tfrac{3}{4}\right)\left(\tfrac{1}{2}\right) = \tfrac{7}{12}. \tag{27}$$

■

Since $\frac{7}{12} > \frac{1}{2}$, this is sufficient to establish the qualitative point that qubit strategies can outperform classical bit strategies—so there is a sense in which a qubit can store more classical information than a bit.

However, $\frac{7}{12} = 0.583...$ is not the best possible performance of qubit strategies. There are better ways for Bob to measure the trine state received from Alice. For example, there is a procedure that attains success probability $\frac{2}{3}\cos^2(\pi/12) = 0.622...$, based on Bob measuring his received trine state with respect to a different orthonormal basis. This improved procedure is left as an exercise.

Exercise 5.1. Show that there exists a qubit strategy that attains worst-case success probability $\frac{2}{3}\cos^2(\pi/12)$.

Moreover, even $\frac{2}{3}\cos^2(\pi/12)$ is not the optimal strategy for measuring an unknown trine state in the worst-case. Can you think of how to do better? In a later section, we will see a somewhat exotic measurement that correctly identifies a trine state with success probability $\frac{2}{3}$.

6 Systems with multiple bits and multiple qubits

Up until now, we have considered systems of a single bit and a single qubit. Let's consider the case of multiple bits and qubits.

6.1 Definitions of n-bit systems and n-qubit systems

Our definitions for bits and qubits extend naturally to n-bit systems and n-qubit systems, by taking 2^n-dimensional vectors instead of 2-dimensional vectors.

For n classical bits, there are 2^n possible values, and a probabilistic state has a probability p_x associated with every n-bit string $x \in \{0,1\}^n$. Of course, since these are probabilities, we have: for all $x \in \{0,1\}^n$, $p_x \geq 0$ and $\sum_{x \in \{0,1\}^n} p_x = 1$. These probabilities constitute a 2^n-dimensional probability vector.

For n quantum bits, there are 2^n amplitudes: $\alpha_x \in \mathbb{C}$, for each $x \in \{0,1\}^n$ (where $\sum_{x \in \{0,1\}^n} |\alpha_x|^2 = 1$). These amplitudes constitute a 2^n-dimensional state vector (which is a unit vector).

Note that, although the focus of attention in quantum information processing is usually on n-qubit systems, it's completely valid to consider systems whose states have dimensions other than powers of 2. For example, a *quantum trit* (*qutrit*) has a 3-dimensional state vector of the form $\alpha_0|0\rangle + \alpha_1|1\rangle + \alpha_2|2\rangle$ (with $|\alpha_0|^2 + |\alpha_1|^2 + |\alpha_2|^2 = 1$).

The set of all probability vectors is a *simplex*, which is illustrated for the case of three dimensions as a triangular region.

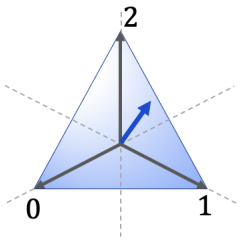

Figure 30: Simplex of all possible 3-dimensional classical (probabilistic) states.

The set of all valid quantum state vectors is a *hypersphere*, which is all points of distance 1 from the origin.

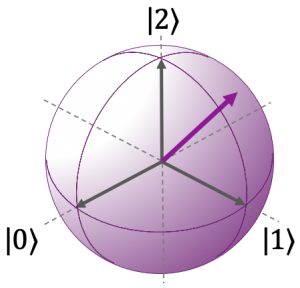

Figure 31: Hypersphere of all possible 3-dimensional quantum states.

There are 2^n (orthonormal) computational basis states, denoted as n-bit strings within kets. For $n = 3$, these states are

$$|000\rangle\,,\ |001\rangle\,,\ |010\rangle\,,\ |011\rangle\,,\ |100\rangle\,,\ |101\rangle\,,\ |110\rangle\,,\ |111\rangle\,. \tag{28}$$

Note that we can write an n-qubit state vector as a linear combination of the 2^n computational basis states, as

$$\sum_{x\in\{0,1\}^n} \alpha_x\,|x\rangle\,, \tag{29}$$

where

$$\sum_{x\in\{0,1\}^n} |\alpha_x|^2 = 1. \tag{30}$$

As with single qubits, what's important is the *operations* that can be performed on them. We'll consider unitary operations and measurements.

Unitary operations are $2^n \times 2^n$ unitary matrices, acting on the 2^n-dimensional state vectors (unitary matrices were defined in section 3.3).

Measurements have 2^n outcomes, corresponding to the 2^n computational basis states. Each basis state outcome occurs with probability the absolute squared of its amplitude. Thus, when a measurement is applied to the state

$$\sum_{x\in\{0,1\}^n} \alpha_x\,|x\rangle\,, \tag{31}$$

what happens is: an *outcome* $x \in \{0,1\}^n$ occurs with probability $|\alpha_x|^2$ and the state of the system changes to the computational basis state $|x\rangle$.

So far, everything is the same as for bits and qubits, except with 2^n dimensions instead of two dimensions. But there's more to it than that. There is structure among subsystems.

6.2 Subsystems of n-bit systems

First, let's consider how subsystems work for the case of a classical n-bit system. It can be viewed as one system (shown here as a rather bloated USB memory stick)

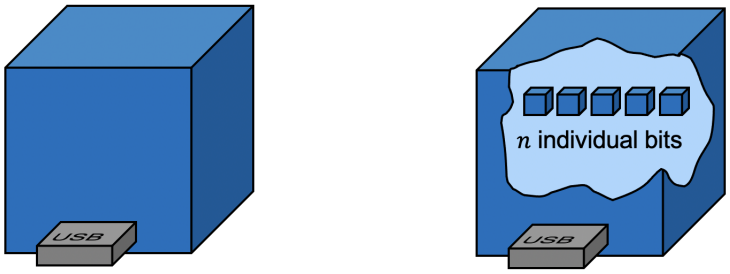

Figure 32: An n-bit system can be viewed as n separate 1-bit systems.

whose state can be described as a 2^n-dimensional probability vector. But we can also view the n-bit system as n separate 1-bit systems. Let's explore that.

We can consider the state of every subset of the n bits. We have a probability vector for the entire system. For three bits it would be this 8-dimensional vector

$$\begin{bmatrix} p_{000} \\ p_{001} \\ p_{010} \\ p_{011} \\ p_{100} \\ p_{101} \\ p_{110} \\ p_{111} \end{bmatrix}. \tag{32}$$

What's the state of the first bit? The probability that the first bit is 0 is the sum of the first four probabilities (all cases where the first bit is 0), and the probability that it's 1 is the sum of the last four probabilities. In this manner, we can deduce the probability vector for the first bit to be

$$\begin{bmatrix} p_{000} + p_{001} + p_{010} + p_{011} \\ p_{100} + p_{101} + p_{110} + p_{111} \end{bmatrix}. \tag{33}$$

By similar reasoning, we can deduce the probability vector for any other subset of the bits. In the language of probability theory, these are called *marginal distributions*.

Also, an operation can act on a subset of the bits. For example, if there are three bits, it makes sense to apply an operation *to the first bit*. For example, think of how applying a NOT operation to the first bit affects the 8-dimensional probability vector in Eq. (32). It permutes the probabilities, resulting in the vector

$$\begin{bmatrix} p_{100} \\ p_{101} \\ p_{110} \\ p_{111} \\ p_{000} \\ p_{001} \\ p_{010} \\ p_{011} \end{bmatrix}. \tag{34}$$

It should be clear that, to apply a NOT operation to the first bit, one only needs to be in possession of the first bit. This operation is local to the first bit.

And operations can be similarly local to various other subsets of the bits. *Dataflow diagrams* are a useful way of illustrating localizations of operations, and their evolution in time. Figure 33 is an example of a dataflow diagram.

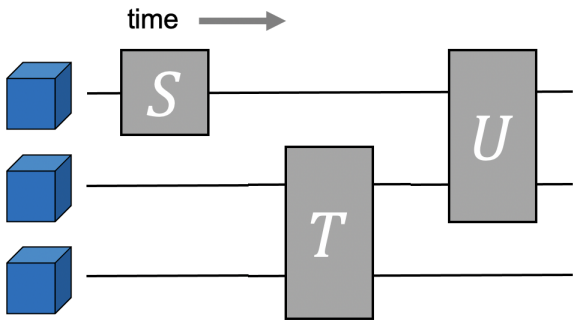

Figure 33: A *dataflow diagram* of a 3-bit system. First, operation S is applied to the first bit. Then operation T is applied jointly to the second and third bits. Finally, operation U is applied to the first and second bits.

6.3 Subsystems of n-qubit systems

Now we consider subsystems in the context of an n-qubit system. An n-qubit system can be viewed as one system (shown in Figure 34 as a bloated quantum USB memory). But it can also be viewed as n separate 1-qubit systems.

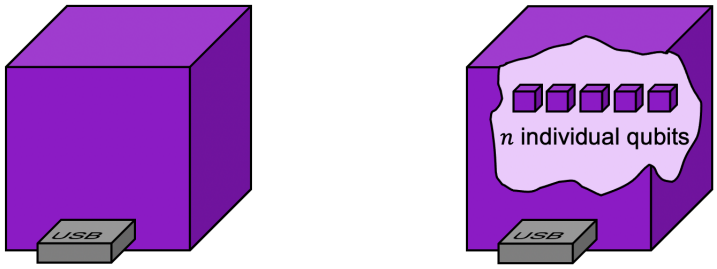

Figure 34: An n-qubit system can be viewed as n separate 1-qubit systems.

Can we consider the state of every subset of the n qubits? Consider a 3-qubit system with 8-dimensional state vector

$$\begin{bmatrix} \alpha_{000} \\ \alpha_{001} \\ \alpha_{010} \\ \alpha_{011} \\ \alpha_{100} \\ \alpha_{101} \\ \alpha_{110} \\ \alpha_{111} \end{bmatrix}. \tag{35}$$

What's the state of the first qubit? Naïvely, we could try summing the first four and the last four amplitudes, as we did for probabilities. But that doesn't work. In fact, for the state vector

$$\begin{bmatrix} \frac{1}{\sqrt{8}} \\ -\frac{1}{\sqrt{8}} \\ \frac{1}{\sqrt{8}} \\ -\frac{1}{\sqrt{8}} \\ \frac{1}{\sqrt{8}} \\ -\frac{1}{\sqrt{8}} \\ \frac{1}{\sqrt{8}} \\ -\frac{1}{\sqrt{8}} \end{bmatrix} \tag{36}$$

this would result in

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{37}$$

and having both amplitudes be zero makes no sense as a one-qubit state vector! Can we do something else instead?

It turns out that the states of subsystems of quantum systems are a bit tricky. We will be able to address this matter in section 29.5 of *Part III: quantum information theory*, using the language of density matrices. For now, it suffices to be aware that: *in some cases, there does not exist a state vector for a subsystem.* In this sense, the larger system must be considered for a quantum state to make sense.

Now, let's consider applying operations to subsets of the qubits. If there are three qubits, does it make sense for a unitary operation to be local to the first qubit? The fact that the first qubit might not even have have a state vector suggests that this is not an entirely trivial matter. But it turns out that there is a fairly straightforward way to make sense of operations that are local to a subset of the qubits—and we'll see how to do this shortly (in section 6.6).

For example, if Alice possesses the first qubit and Bob the last two qubits then Alice can perform an operation on her qubit, without touching Bob's qubits. And operations can be similarly localized to various other subsets of the qubits. *Quantum dataflow diagrams* are a useful way of illustrating localizations of operations, and their evolution in time. They are commonly called *quantum circuits*.

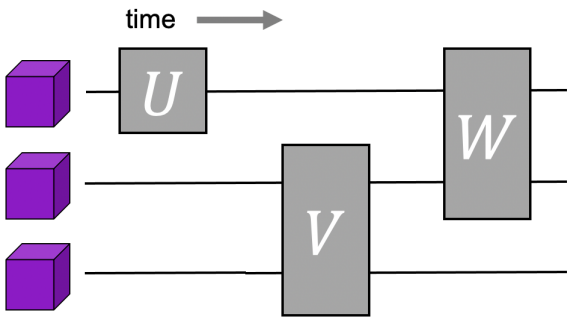

Figure 35: A *quantum circuit* of a 3-qubit system. First, unitary operation U is applied to the first qubit. Then unitary operation V is appied jointly to the second and third qubit. Finally, unitary operation W is applied jointly to the first and second qubits.

There are two ways of viewing a quantum circuit:

- One way is that the lines are wires and the qubits flow along the wires from left to right, and are transformed when the qubits pass through the boxes, which are called *gates*.
- Another way of viewing a quantum circuit is that the qubits stay put and the

horizontal axis only represents time.

Quantum circuits are a very useful way of representing quantum information processes, and you'll be seeing a lot of them.

6.4 Product states

Let's return to the issue of quantum states of subsystems. Remember that multi-qubit state vectors do not always have meaningful state vectors for their subsystems.

However, we can build some quantum state "bottom-up," by starting with the states of the subsystems. For example, consider two qubits in these specific states,

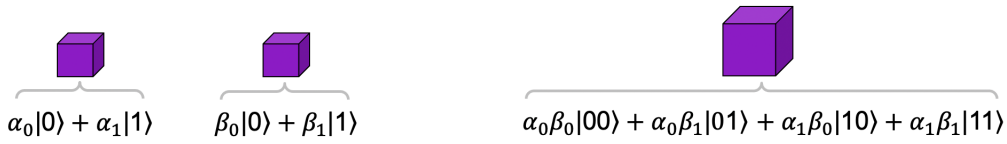

Figure 36: Two separate qubit state vectors can be translated into a 2-qubit state vector.

with amplitudes α_0 and α_1 for the first qubit, and β_0 and β_1 for the second qubit. We can choose to consider these two qubits as two separate systems, or as one 2-qubit system, whose state is a 4-dimensional vector. What is the four-dimensional vector? It's *defined* to be the *tensor product* $\otimes$ of the two 2-dimensional vectors. An intuitive way of thinking about this tensor product is to "expand the product" of the two superpositions, which is

$$(\alpha_0 |0\rangle + \alpha_1 |1\rangle) \otimes (\beta_0 |0\rangle + \beta_1 |1\rangle) = \alpha_0\beta_0 |00\rangle + \alpha_0\beta_1 |01\rangle + \alpha_1\beta_0 |10\rangle + \alpha_1\beta_1 |11\rangle . \tag{38}$$

This definition of the tensor product is equivalent to

$$\begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} \otimes \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} \alpha_0\beta_0 \\ \alpha_0\beta_1 \\ \alpha_1\beta_0 \\ \alpha_1\beta_1 \end{bmatrix} . \tag{39}$$

Note that this is similar to the way that probability distributions of independent systems are combined to yield product distributions.

We now define the *tensor product* for arbitrary matrices (the case of d-dimensional column vectors occurs as the special case of $d \times 1$ matrices).

Definition 6.1. Let A and B be $n \times m$ and $k \times \ell$ matrices (respectively):

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{bmatrix} \qquad B = \begin{bmatrix} B_{11} & B_{12} & \cdots & B_{1\ell} \\ B_{21} & B_{22} & \cdots & B_{2\ell} \\ \vdots & \vdots & \ddots & \vdots \\ B_{k1} & B_{n2} & \cdots & B_{k\ell} \end{bmatrix}. \tag{40}$$

The *tensor product* of A and B (also called the *Kronecker product*) is defined as

$$A \otimes B = \left[\begin{array}{c|c|c|c} A_{11}B & A_{12}B & \cdots & A_{1m}B \\ \hline A_{21}B & A_{22}B & \cdots & A_{2m}B \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline A_{n1}B & A_{n2}B & \cdots & A_{nm}B \end{array}\right], \tag{41}$$

where each $A_{ij}B$ denotes a $k \times \ell$ block consisting of all entries of B multiplied by A_{ij}. Note that $A \otimes B$ is an $nk \times m\ell$ matrix.

Definition 6.2. If one system is in state $|\psi\rangle$ and another system is in state $|\phi\rangle$, then the state of the joint system is the *product state* $|\phi\rangle \otimes |\psi\rangle$.

Now a few words about notation for product states. Frequently $|\phi\rangle \otimes |\psi\rangle$ is abbreviated to $|\phi\rangle\,|\psi\rangle$. Also, for computational basis states, $|a\rangle$ and $|b\rangle$ (where $a \in \{0,1\}^n$ and $b \in \{0,1\}^m$), we have these equivalent notations: $|a\rangle \otimes |b\rangle = |a\rangle\,|b\rangle = |ab\rangle$. For example, $|0\rangle \otimes |0\rangle \otimes |1\rangle = |0\rangle\,|0\rangle\,|1\rangle = |001\rangle$.

Exercise 6.1 (straightforward, but one case is a trick question)**.** In each case, express the 2-qubit state as a product of two 1-qubit states:

$$\tfrac{1}{2}|00\rangle + \tfrac{1}{2}|01\rangle + \tfrac{1}{2}|10\rangle + \tfrac{1}{2}|11\rangle \tag{42}$$

$$\tfrac{1}{2}|00\rangle - \tfrac{1}{2}|01\rangle - \tfrac{1}{2}|10\rangle + \tfrac{1}{2}|11\rangle \tag{43}$$

$$\tfrac{1}{4}|00\rangle + \tfrac{\sqrt{3}}{4}|01\rangle + \tfrac{\sqrt{3}}{4}|10\rangle + \tfrac{3}{4}|11\rangle \tag{44}$$

$$\tfrac{1}{\sqrt{2}}|00\rangle + \tfrac{1}{\sqrt{2}}|11\rangle\,. \tag{45}$$

The first three cases are straightforward. If you tried to work out the third case, you probably realized that there is no solution! The last state cannot be expressed as a tensor product. It is one of those states (mentioned in section 6.3) whose individual qubits do not have state vectors.

Exercise 6.2 (fairly straightforward)**.** Prove that the state vector $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ cannot be written as the tensor product of two one qubit state vectors.

The state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ is an example of an *entangled* state. We'll see that two qubits in such a state can behave in interesting ways. It's especially interesting when the two qubits are physically in separate locations, say one is in Alice's lab and one is in Bob's lab.

6.5 Notation for explicitly referring to individual qubits

Consider the state $|000\rangle + |110\rangle$ (using the normalization convention in section 3.2). Since we can write this state as $\Big(|00\rangle + |11\rangle\Big) \otimes |0\rangle$, it's reasonable to say that the first qubit is entangled with the second qubit and also that the first two qubits are in a product state with the third qubit. Now, what about the state $|000\rangle + |101\rangle$? The first and third qubit are in a product state with the second qubit; however, $|000\rangle + |101\rangle$ cannot be written as a tensor product using our notation (go ahead and try).

A remedy is to extend our notation. The notion of a "first qubit," "second qubit" (etc) are just arbitrary ways of referring to individual qubits and do not imply anything about a physical layout of the qubits.

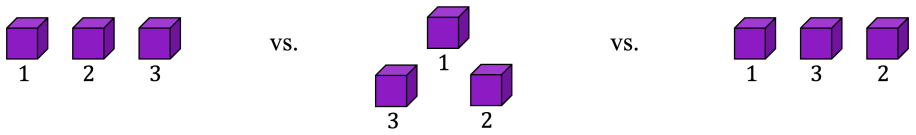

Figure 37: The ket notation does not imply any particular physical layout of the qubits.

We can subscript the qubits with labels to enable us to write kets in any order, without ambiguity. For example, we can write $|1\rangle_1 |0\rangle_2 |1\rangle_3 = |1\rangle_1 |1\rangle_3 |0\rangle_2$ and then

$$|0\rangle_1 |0\rangle_2 |0\rangle_3 + |1\rangle_1 |0\rangle_2 |1\rangle_3 = \Big(|0\rangle_1 |0\rangle_3 + |1\rangle_1 |1\rangle_3\Big) \otimes |0\rangle_2 . \tag{46}$$

To reduce clutter, we can use reasonable abbreviations to simplify this to

$$|000\rangle_{1,2,3} + |101\rangle_{1,2,3} = \Big(|00\rangle_{1,3} + |11\rangle_{1,3}\Big) \otimes |0\rangle_2 . \tag{47}$$

Now suppose that Alice has two qubits in her lab and Bob has two qubits in his lab and their first qubits are entangled in state $|00\rangle + |11\rangle$ as well as their second qubits. Writing $\Big(|00\rangle + |11\rangle\Big)\Big(|00\rangle + |11\rangle\Big)$ is ambiguous because it can be parsed as Alice and having entanglement *between her two qubits*, in a product state with Bob's

two qubits. We could disambiguate by using subscripts A_1, A_2, B_1, B_2 for the four qubits. A simpler, less cluttered, notation that's sufficiently clear in most contexts is to use A for both of Alice's qubits and B for both of Bob's qubits. Then we can write

$$\begin{aligned}
&\left(|00\rangle_{AB} + |11\rangle_{AB}\right) \otimes \left(|00\rangle_{AB} + |11\rangle_{AB}\right) && (48)\\
&\quad = |00\rangle_{AB}\,|00\rangle_{AB} + |00\rangle_{AB}\,|11\rangle_{AB} + |11\rangle_{A}\,|00\rangle_{B} + |11\rangle_{A}\,|11\rangle_{B} && (49)\\
&\quad = |00\rangle_{A}\,|00\rangle_{B} + |01\rangle_{A}\,|01\rangle_{B} + |10\rangle_{A}\,|10\rangle_{B} + |11\rangle_{A}\,|11\rangle_{B}\,. && (50)
\end{aligned}$$

6.6 Local unitary operations

Now, let's consider the matter of the scope of unitary operations. Suppose that there are two qubits and we want to apply a 1-qubit unitary operation

$$U = \begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix} \tag{51}$$

to the second qubit (and do nothing to the first qubit), as illustrated in figure 38.

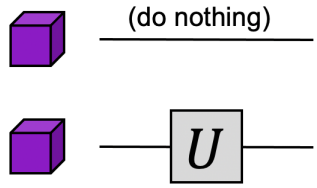

Figure 38: Circuit diagram of a 1-qubit unitary U acting on the second qubit of a 2-qubit system.

What is the 4×4 unitary matrix acting on the 2-qubit system that expresses this?

If the individual qubits happen to be in computational basis states then it's reasonable that the first state does not change and the second state is acted on by U, so the 4×4 unitary must have the property that

$$\begin{aligned}
|0\rangle\,|0\rangle &\mapsto |0\rangle\,U\,|0\rangle && (52)\\
|0\rangle\,|1\rangle &\mapsto |0\rangle\,U\,|1\rangle && (53)\\
|1\rangle\,|0\rangle &\mapsto |1\rangle\,U\,|0\rangle && (54)\\
|1\rangle\,|1\rangle &\mapsto |1\rangle\,U\,|1\rangle\,. && (55)
\end{aligned}$$

Now, if we have a 4×4 unitary matrix with this effect on the basis states then, by linearity, it must be

$$\begin{bmatrix} u_{00} & u_{01} & 0 & 0 \\ u_{10} & u_{11} & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{bmatrix}. \tag{56}$$

This is what we will take as the *definition* of doing nothing to the first qubit and applying U to the second qubit.

Notice that, by this definition, it makes perfect sense to apply U to the second qubit of *any* 2-qubit system, even one in an entangled state like

$$\tfrac{1}{\sqrt{2}} |00\rangle + \tfrac{1}{\sqrt{2}} |11\rangle, \tag{57}$$

where the second qubit of the state does not even have a state vector! Whatever the 2-qubit state is, it's a 4-dimensional vector, and it makes sense to multiply that vector by the matrix in Eq. (56).

Interestingly, the matrix in Eq. (56) can be expressed succinctly as

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix} = I \otimes U, \tag{58}$$

where the operation $\otimes$ (the tensor product) is defined in Definition 6.1. Here's a question to consider:

Exercise 6.3 (straightforward)**.** What is the 4×4 unitary corresponding to applying U to the *first* qubit and doing nothing to the *second* qubit?

We've discussed 1-qubit unitary operations in 2-qubit systems. This generalizes naturally to more qubits. For example, when there are $n+m$ qubits and U is applied to the last m qubits, the resulting $2^{n+m} \times 2^{n+m}$ matrix is $I \otimes U$, where I is the $2^n \times 2^n$

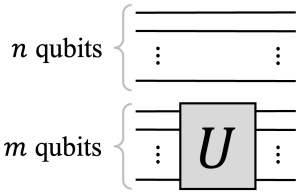

Figure 39: Circuit for U applied to the last m qubits of an $(n+m)$-qubit system.

identity matrix. Also, if a unitary V is applied to the first n qubits, this is expressed as $V \otimes I$, where I is the $2^m \times 2^m$ identity matrix.

Furthermore, if U and V act on separate qubits then the two operations commute,

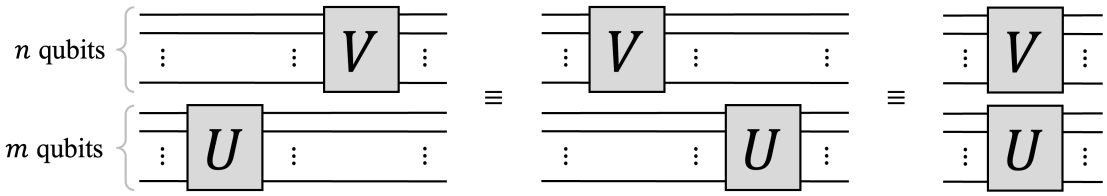

Figure 40: Two local unitaries acting on separate qubits can be done in any order.

and can be applied in parallel, as illustrated in figure 40. The combined operation of applying V to the first n qubits and U to the last m qubits corresponds to the $2^{n+m} \times 2^{n+m}$ matrix $V \otimes U$. The equivalences in figure 40 are a consequence of a multiplicative property of tensors that is explained in the next subsection (section 6.7).

Now, consider the circuit of the form illustrated in figure 41. This circuit can be

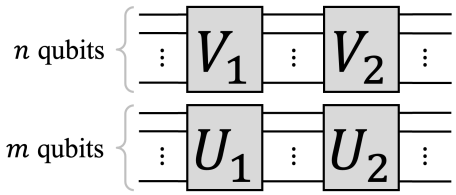

Figure 41: This circuit can be interpreted in two ways (and they are equivalent).

interpreted in two ways:

- Perform V_2V_1 on the first n qubits and, in parallel, perform U_2U_1 on the last m qubits. The net effect of this process is $\big(V_2V_1\big) \otimes \big(U_2U_1\big)$.
- Perform $V_1 \otimes U_1$ on the $n+m$ qubits followed by $V_2 \otimes U_2$. The net effect of this process is $\big(V_2 \otimes U_2\big)\big(V_1 \otimes U_1\big)$.

The two interpretations are equivalent—so they are both are valid—because

$$\big(V_2 \otimes U_2\big)\big(V_1 \otimes U_1\big) = \big(V_2V_1\big) \otimes \big(U_2U_1\big). \tag{59}$$

Eq. (59) is a special case of a multiplicative property of tensors that is explained in the next section.

6.7 Multiplicative property of tensors

The following lemma is more general than Eq. (59), because it holds for *any* matrices (that need not even be square) as long as the dimensions are such that the matrix products make sense.

Lemma 6.1 (multiplicative property of tensors)**.** *Let A be a $c_1 \times d_1$ matrix and C be an $d_1 \times r_1$ matrix (so the matrix product AC makes sense). Let B be an $c_2 \times d_2$ matrix and D be an $d_2 \times r_2$ matrix (so the matrix product BD makes sense). Then*

$$\Big(A \otimes B\Big)\Big(C \otimes D\Big) = \Big(AC\Big) \otimes \Big(BD\Big). \tag{60}$$

Proof. The equations

$$\Big(A \otimes I\Big)\Big(I \otimes B\Big) = A \otimes B = \Big(I \otimes B\Big)\Big(A \otimes I\Big) \tag{61}$$
$$\Big(A \otimes I\Big)\Big(C \otimes I\Big) = \Big(AC\Big) \otimes I \tag{62}$$
$$\Big(I \otimes B\Big)\Big(I \otimes D\Big) = I \otimes \Big(BD\Big) \tag{63}$$

are all fairly easy to verify directly from definition 6.1 (the definition of the tensor product).

Using the above equations,

$$\begin{aligned}
\Big(A \otimes B\Big)\Big(C \otimes D\Big) &= \Big(A \otimes I\Big)(I \otimes B\Big)\Big(C \otimes I\Big)\Big(I \otimes D\Big) && \text{(by Eq. (61))} && (64)\\
&= \Big(A \otimes I\Big)\Big(C \otimes I\Big)(I \otimes B\Big)\Big(I \otimes D\Big) && \text{(by Eq. (61))} && (65)\\
&= \Big((AC) \otimes I\Big)(I \otimes (BD)\Big) && \text{(by Eqns. (62)(63))} && (66)\\
&= \Big(AC\Big) \otimes \Big(BD\Big). && \text{(by Eq. (61))} && (67)
\end{aligned}$$

■

6.8 Controlled-U gates

Now, I'd like to show you a unitary operation called a *controlled-U gate*, where U itself can be any unitary operation.

For example, consider the case where U is the 1-qubit unitary operation

$$U = \begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix}. \tag{68}$$

The notation for the associated controlled-U gate in circuit diagrams is the following.

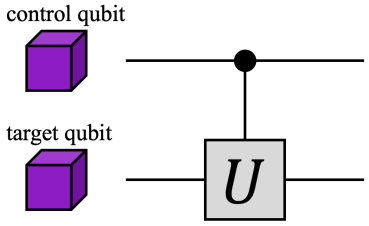

Figure 42: Notation for controlled-U gate.

where U drawn as "acting" on the *target qubit* and with a vertical "wire" from the *control qubit* to U.

If the control qubit is in state $|0\rangle$ then nothing happens. And, if the control qubit is in state $|1\rangle$ then U gets applied to the target qubit. This gate has the following effect on the four computational basis states:

$$|0\rangle\,|0\rangle \mapsto |0\rangle\,|0\rangle \tag{69}$$
$$|0\rangle\,|1\rangle \mapsto |0\rangle\,|1\rangle \tag{70}$$
$$|1\rangle\,|0\rangle \mapsto |1\rangle\,U\,|0\rangle \tag{71}$$
$$|1\rangle\,|1\rangle \mapsto |1\rangle\,U\,|1\rangle\,. \tag{72}$$

By linearity, we can deduce from this that the 4×4 matrix of this controlled-U gate is the matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & u_{00} & u_{01} \\ 0 & 0 & u_{10} & u_{11} \end{bmatrix}. \tag{73}$$

The matrix in Eq. (73) is the *definition* of the controlled-U gate acting on two qubits, which applies for any 2-qubit input state—including states where the control qubit is not in a computational basis state.

It is important to stress that, when the control qubit is not in a computational basis state, it is *not valid* to think of the effect of the controlled-U gate as

$$\begin{cases} \text{apply } I & \text{if the control qubit is in state } |0\rangle \\ \text{apply } U & \text{if the control qubit is in state } |1\rangle. \end{cases} \tag{74}$$

Nevertheless, the controlled-U gate makes sense for all input states. Whatever the 2-qubit state is, it's a 4-dimensional vector, and it makes sense to multiply that vector by the matrix in Eq. (73).

Exercise 6.4 (worth thinking about)**.** Does there exist a controlled-U gate that changes the state of its *control* qubit? To make "the state of the control qubit" clear, let's restrict to states where the input state and output state are both product states. What does your intuition say?

The aforementioned definition of a controlled-U gate was for the case where the first qubit is the control qubit and the second qubit is the target qubit. There is natural corresponding definition for the case where the orientation is inverted (where second qubit is the control qubit and the first qubit is the target qubit).

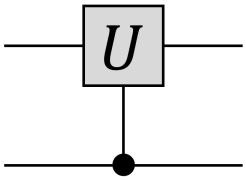

Figure 43: A controlled-U gate with inverted control/target orientation.

Exercise 6.5. Consider an inverted control-U gate, where the second qubit is the control and the first qubit is the target. Based on the above explanations, how should the 4×4 matrix be defined for this (analogous to the matrix in Eq. (73))?

Finally, a controlled-U gate can be defined for any n-qubit unitary U. The controlled-U gate is an $(n+1)$-qubit gate, where the additional qubit is the control qubit.

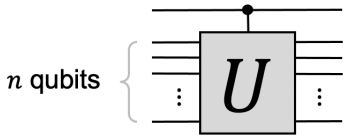

Figure 44: Notation for a controlled-U gate for an n-qubit U.

If the control qubit is the first qubit then the controlled-U gate is defined as the $2^{n+1} \times 2^{n+1}$ matrix

$$\left[\begin{array}{c|c} I & 0 \\ \hline 0 & U \end{array}\right], \tag{75}$$

where I and U are both $2^n \times 2^n$ blocks.

6.9 Controlled-NOT gate (a.k.a. CNOT)

Here we consider the controlled-U gate in the case where

$$U = X = \mathsf{NOT} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}. \tag{76}$$

This 2-qubit gate is commonly referred to as the CNOT gate (controlled-NOT). It has interesting properties and occurs very frequently in the theory of quantum information processing. There is special notation for this gate, shown in figure 45.

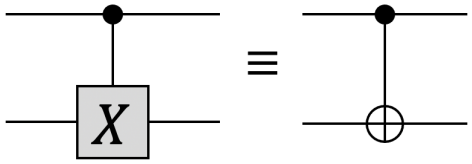

Figure 45: Controlled-NOT gate (two different notations).

To understand where the notation comes from, consider what happens when the inputs are computational basis states. Let the inputs be $|a\rangle$ and $|b\rangle$, where $a, b \in \{0, 1\}$. For these input states, the output states are $|a\rangle$ and $|a \oplus b\rangle$. The sym-

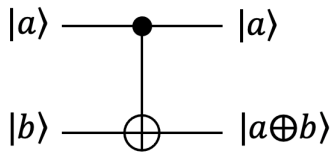

Figure 46: Action of CNOT gate on the computational basis states ($a, b \in \{0, 1\}$).

bol $\oplus$ is the binary exclusive-OR operation (a.k.a. XOR). If you haven't seen the $\oplus$ operation before, here's a table of its values, and a comparison with the values of $\vee$ (the standard OR).

	XOR	OR
ab	$a \oplus b$	$a \vee b$
00	0	0
01	1	1
10	1	1
11	0	1

The value of $a \oplus b$ is 1 *and only if* one of the two input bits are 1, *but not both*; whereas, $a \vee b$ is 1 also in the case where both a and b are 1. Another, altogether different, way of thinking about the $\oplus$ operation is that it is the sum of the two bits in modulo 2 arithmetic. The way that the symbol $\oplus$ is embedded into the gate symbol in figure 46 is suggestive of what it does.

The above discussion of the CNOT gate is for computational basis states. The *definition* of the CNOT gates is given by the 4×4 unitary matrix in Eq. (73)

$$\mathsf{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{77}$$

This operation can be applied to *any* 2-qubit state—independent of any intuitive picture that's based on the very special case of computational basis states.

Remember, in Exercise 6.4, I asked a question about whether there is a controlled-U gate that can change the state of its control qubit? What did you decide?

Feel free to think more about this before looking at the next page ...

The answer might surprise you: for some input states to the CNOT gate, the control qubit actually changes! Recall the states

$$|+\rangle = \tfrac{1}{\sqrt{2}}|0\rangle + \tfrac{1}{\sqrt{2}}|1\rangle \tag{78}$$

$$|-\rangle = \tfrac{1}{\sqrt{2}}|0\rangle - \tfrac{1}{\sqrt{2}}|1\rangle\,. \tag{79}$$

(first defined in section 4). Suppose the control qubit is set to $|+\rangle$ and the target qubit is set to $|-\rangle$ and then the CNOT gate is applied. It can be verified by a calculation that the output qubits are both in state $|-\rangle$. So, for this input, the control qubit

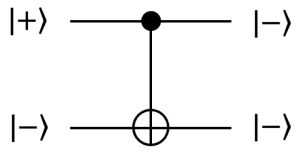

Figure 47: Example where CNOT gate modifies the state of the control qubit.

changes state, from $|+\rangle$ to $|-\rangle$. And recall that, as we saw in section 4, $|+\rangle$ and $|-\rangle$ are certainly different states—they're orthogonal and perfectly distinguishable.

Exercise 6.6 (straightforward)**.** Verify that $\mathsf{CNOT}\big(|+\rangle \otimes |-\rangle\big) = |-\rangle \otimes |-\rangle$.

The CNOT gate has several other interesting properties. One other property concerns the simulation of *other* controlled-U gates, for different unitary operations U, other than the X gate. Suppose that we have the capability of performing CNOT gates plus all one-qubit unitary operations—and that's all. Then, can we construct circuits with these gates that implement other controlled-U gates? Let's start by considering the the controlled-R_θ, where R_θ is the rotation by angle θ

$$R_\theta = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}. \tag{80}$$

How do we approach this? Well, we can guess a few simple forms that the circuit might take. Consider a quantum circuit of this form.

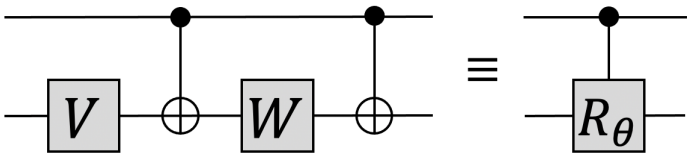

Figure 48: Simulating a controlled-U gate from CNOT gates and one-qubit gates.

Do there exist unitaries V and W such that this circuit simulates the controlled-R_θ? The answer is yes, and I leave this as an exercise.

Exercise 6.7 (fairly straightforward)**.** Find 1-qubit unitary operations V and W such that the circuit on the left side of figure 48 performs the same unitary operation as the controlled-R_θ. (Hint: consider setting V and W to rotation matrices, with carefully chosen angles.)

Exercise 6.7 is a good starting point towards this more challenging problem:

Exercise 6.8 (challenging)**.** Show how to simulate a controlled-U operation for any 1-qubit unitary U by a circuit consisting of only CNOT and 1-qubit gates. Note that the form of the simulating circuit need not be the same as the left side of figure 48. (Hint: begin by considering the case where U has determinant 1.)

7 Superdense coding

This section is about an interesting communication feat that is possible with qubits called *superdense coding*.[8] What makes superdense coding interesting is not its power (the feat of communicating two bits is not very exciting); rather, superdense coding shows a sense in which quantum information fundamentally differs from classical information. In particular, entangled states can be manipulated in interesting ways.

7.1 Prelude to superdense coding

Suppose that Alice wants to convey two classical bits to Bob by sending only one classical bit.

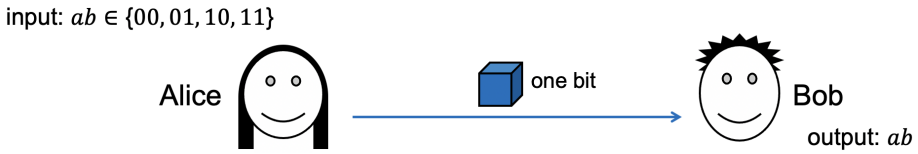

Figure 49: Scenario for Alice conveying two bits ab to Bob by sending just one bit (the best strategy succeeds with probability $\frac{1}{2}$).

The precise scenario is that Alice receives her two bits, $a, b \in \{0, 1\}$ as input and then she somehow creates a 1-bit message to send to Bob, who is somehow supposed to determine both a and b from the bit that he receives from Alice. It should be clear that this is impossible to accomplish perfectly. The highest success probability possible is $\frac{1}{2}$, and this is obtained by the simple strategy where Alice just sends a to Bob and then Bob outputs a and randomly guesses the value of b. This strategy has success probability $\frac{1}{2}$ in the average-case as well as in the worst-case.

What if Alice can send a *qubit* to Bob?

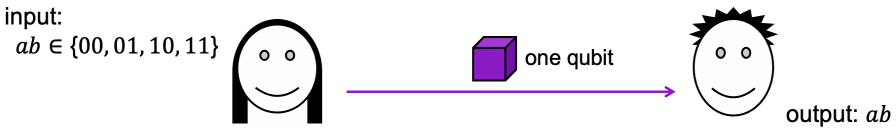

Figure 50: Scenario where Alice can send a qubit (the best success probability is $\frac{1}{2}$).

It turns out that this does not help: the best success probability is still $\frac{1}{2}$. We don't prove this here (it's a consequence of a result of Nayak[7]).

[8] C. H. Bennett and S. J. Wiesner (1992). "Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states." *Phys. Rev. Lett.*, **69**(20): 2881–2884.

Now, let's add a twist. What if we allow Bob to send a bit to Alice before Alice sends her bit to him?

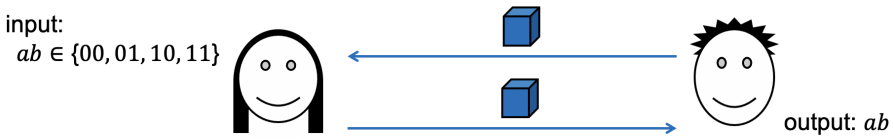

Figure 51: Scenario where Bob can send a bit to Alice and then Alice can send a bit to Bob (the best possible success probability is $\frac{1}{2}$).

To be clear, the scenario (depicted in figure 51) is the following:

1. Alice receives her two bits, $a, b \in \{0, 1\}$ as input.
2. Bob sends a bit to Alice.
3. Alice sends a bit to Bob.
4. Then Bob outputs two bits (and this is *successful* if his output bits are ab).

That extra bit of communication from Bob to Alice does not help. The best possible success probability is still $\frac{1}{2}$. Intuitively, this is because the flow of information is in the wrong direction. How does Bob sending a bit to Alice provide *him* with any more information? To be sure that there isn't some subtle way that Bob's message helps, we would need to think about this carefully. But let's just accept, without proof, that the best possible success probability is $\frac{1}{2}$.

In fact, if Bob sends a bit the wrong way and then Alice sends a *qubit* to Bob, even that does not help: the best possible success probability is *still* $\frac{1}{2}$.

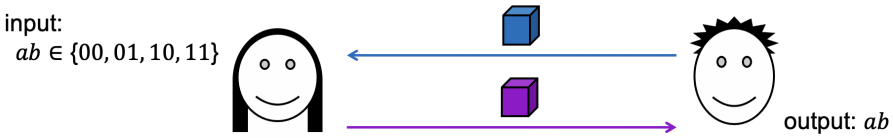

Figure 52: Scenario where Bob can send a bit to Alice and then Alice can send a qubit to Bob (the best possible success probability is $\frac{1}{2}$).

These examples seem to indicate that *messages sent in the wrong direction are of no use*. We will see that superdense coding violates this intuition. In superdense coding, Bob first sends a *qubit* to Alice and then Alice sends a *qubit* to Bob—and Bob's message actually makes a difference: the protocol always succeeds!

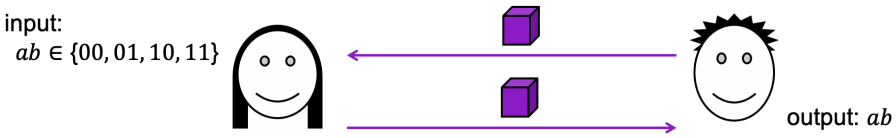

Figure 53: Scenario where Bob can send a qubit to Alice and then Alice can send a qubit to Bob (the *superdense coding* protocol always succeeds at this).

The scenario is that:

1. Alice receives her two bits, $a, b \in \{0, 1\}$ as input.
2. Bob sends a qubit to Alice.
3. Alice sends a qubit to Bob.
4. Then Bob outputs two bits (and this is *successful* if his output bits are ab).

We'll see a communication protocol of this form where Bob always outouts ab correctly. Sending a *bit* in the wrong direction does not help but, somehow, sending a *qubit* in the wrong direction does help!

7.2 How superdense coding works

Let's begin with a description of the protocol for superdense coding. It is the following three steps.

1. Bob creates the entangled two-qubit state

$$\tfrac{1}{\sqrt{2}}|00\rangle + \tfrac{1}{\sqrt{2}}|11\rangle \tag{81}$$

   then he sends the first qubit to Alice (and he keeps the second qubit). So, at this point, Alice and Bob each possess one qubit of this 2-qubit state.

2. Alice has her two input bits a and b and the qubit that she received from Bob. She performs the following procedure:
   2.1 If $a = 1$ apply X to the qubit (where $X = \left[\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right]$).
   2.2 If $b = 1$ apply Z to the qubit (where $Z = \left[\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right]$).
   In summary, Alice applies $Z^b X^a$ to the qubit in her possession. Then she sends her qubit to Bob.

3. At this point Bob is in possession of both qubits again. He applies the circuit in figure 54 to the two qubits and measures in the computational basis.

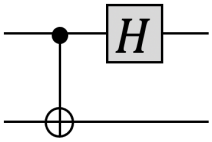

Figure 54

The outcome of Bob's measurement is two bits, which is his output.

Now, let's analyze how this protocol works. In step 2, Alice's operations on the first qubit changes the 2-qubit state in the following way:

$$\begin{cases} \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle & \text{if } ab = 00 \\ \frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle & \text{if } ab = 01 \\ \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle & \text{if } ab = 10 \\ \frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|10\rangle & \text{if } ab = 11. \end{cases} \tag{82}$$

There's something interesting about these four states: they are orthogonal to each other! They are an orthonormal basis for the 4-dimensional state space associated with two qubits. This is called the *Bell basis* (named after John Bell).

What Bob does in step 3 is measure the two qubits *in the Bell basis*. This is accomplished by Bob first applying the unitary operation specified by the circuit in figure 54 and then measuring in the computational basis. The effect of the unitary operation on the four Bell states is shown in the following table (where we are omitting the $\frac{1}{\sqrt{2}}$ factors to reduce clutter, following the normalization convention in section 3.2).

input	output
$\|00\rangle + \|11\rangle$	$\|00\rangle$
$\|00\rangle - \|11\rangle$	$\|01\rangle$
$\|01\rangle + \|10\rangle$	$\|10\rangle$
$\|01\rangle - \|10\rangle$	$\|11\rangle$

Therefore, when Bob measures in the computational basis, he recovers the bits ab, as required.

So that's how superdense coding works. It makes use of an interesting property of the Bell basis, where, in step 2, Alice applies an operation to just one of the two qubits (the one in her possession) but by doing so she manages to change the state to any of the four Bell basis states. That step wouldn't work if the computational basis were used: Alice could then manipulate the state of the first qubit but she couldn't do anything to the second qubit, which is in Bob's possession. And there is no way to do this using classical bits.

8 Projective and local measurements

So far, our notion of measurement has been with respect to some orthonormal basis, the measurement causes the state to collapse to one of the elements of the orthonormal basis. Here we broaden our notion of measurement to include measurements that yield *less* information than this, in return for a benefit: that the measurement can be less destructive to the quantum state. An example of this is a *local* measurement, that measures a subset of a set of qubits.

8.1 Projective measurements

First, I'll show you a more general notion of measurement than anything we've discussed so far, which we call a *projective measurement*. We need at least three dimensional quantum state vectors to illustrate this kind of measurement.

We have considered 2-qubit systems, whose state vectors are 4-dimensional. But let's start with 3-dimensional quantum systems, where the space of states is easier to visualize. For a quantum trit (or *qutrit*) there are three computational basis states, called $|0\rangle$, $|1\rangle$, and $|2\rangle$.

The most general measurements that we have seen so far are based on an orthonormal basis $|\phi_0\rangle, |\phi_1\rangle, |\phi_2\rangle$ and they act as follows: they project the state to one of the computational basis states, where the probability of projecting to each such basis state is the projection length squared. Since we can perform unitary operations before and after the measurement, we lose no generality by focusing on measurements with respect to the computational basis.

The outcome of the measurement consists of two parts:

- Classical information indicating which basis state occurred—for qutrits, that's 0, 1, or 2—which we can imagine is what we see on the screen of our measurement device (the qutrit analogue of the device in Figure 15).
- And there is also a residual (or collapsed) quantum state, which would be $|0\rangle$, $|1\rangle$, or $|2\rangle$.

An equivalent way of viewing this is that there are three orthogonal one-dimensional subspaces (the span of $|0\rangle$, the span of $|1\rangle$, and the span of $|2\rangle$), and the state has an orthogonal projection onto each subspace, and the square of the length of that projection determines the probability of that outcome.

A *projective measurement* is like this, except that the orthogonal subspaces need not be one-dimensional. For example, for qutrits, consider these two subspaces:

- The horizontal plane spanned by $|0\rangle$ and $|1\rangle$, which is two-dimensional.
- The vertical line spanned by $|2\rangle$, which is one-dimensional.

These two subspaces (illustrated on the right side of figure 55) are orthogonal to each other and, together, they span the entire space.

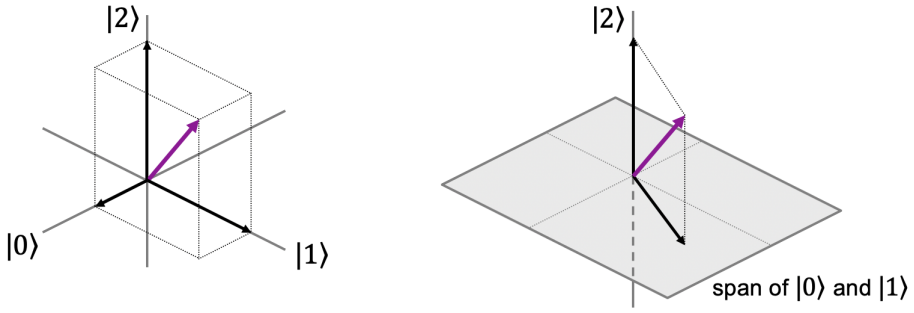

Figure 55: A geometric view of a standard qutrit measurement (left) and an example of a *projective* qutrit measurement (right).

The definition of the projective measurement with respect to these subspaces is as follows. Any quantum state vector has an orthogonal projection on each subspace. The squares of the lengths of these projections sum to 1. The measurement process produces: a *classical* outcome, indicating which space was collapsed to; and a *residual (collapsed) state*, which is the original state projected into one of the subspaces.

For the subspaces on the right side of figure 55 (call them "plane" and "line") the result of measuring the state $\alpha_0 |0\rangle + \alpha_1 |1\rangle + \alpha_2 |2\rangle$ is:

$$\begin{cases} \text{"plane" and residual state } \alpha_0 |0\rangle + \alpha_1 |1\rangle & \text{with probability } |\alpha_0|^2 + |\alpha_1|^2 \\ \text{"line" and residual state } \alpha_2 |2\rangle & \text{with probability } |\alpha_2|^2 \end{cases} \tag{83}$$

(where in both cases the residual state is assumed to be normalized, following our normalization convention for quantum states in section 3.2). In the case of the first outcome, the residual state can still be an interesting quantum state in the sense that it's a superposition of basis states $|0\rangle$ and $|1\rangle$.

I have explained the above projective measurement by appealing to your geometric intuition from figure 55. A more mathematically formal definition of the projective measurement is in terms of matrices that are projectors.

Definition 8.1. A $d \times d$ matrix Π is an *(orthogonal) projector* if $\Pi = \Pi^*$ and $\Pi^2 = \Pi$.

Definition 8.2. A sequence of (orthogonal) projectors $\Pi_0, \Pi_1, \ldots, \Pi_{m-1}$ is *complete* if $\Pi_0 + \Pi_1 + \cdots + \Pi_{m-1} = I$.

The projective measurement illustrated on the right side of figure 55 and defined in Eq. (83) can also be defined in terms of the projectors

$$\Pi_{\text{plane}} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \Pi_{\text{line}} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{84}$$

For $|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle + \alpha_2 |2\rangle$,

$$\Pi_{\text{plane}} |\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle \tag{85}$$

$$\Pi_{\text{line}} |\psi\rangle = \alpha_2 |2\rangle , \tag{86}$$

and the outcome probabilities in Eq. (83) are

$$\langle\psi| \Pi_{\text{plane}} |\psi\rangle \tag{87}$$

$$\langle\psi| \Pi_{\text{line}} |\psi\rangle . \tag{88}$$

Definition 8.3 (projective measurement)**.** For complete $d \times d$ (orthogonal) projectors, $\Pi_0, \Pi_1, \ldots, \Pi_{m-1}$, the associated *projective measurement* is the following process. For any d-dimensional state $|\psi\rangle$, the classical outcome is $k \in \{0, 1, \ldots, m-1\}$, where

$$\Pr[\text{outcome is } k] = \langle\psi| \Pi_k |\psi\rangle , \tag{89}$$

and the corresponding residual state is $\Pi_k |\psi\rangle$, which in normalized form is

$$\frac{\Pi_k |\psi\rangle}{\sqrt{\langle\psi| \Pi_k |\psi\rangle}}. \tag{90}$$

There is some arbitrariness in the choice of labels for the projectors; for m projectors, $\{0, 1, \ldots, m-1\}$ is the default choice.

A word of caution: there exist *non-orthogonal* projectors that satisfy $P^2 = P$; but for which $P^* \neq P$. An example of such a matrix is

$$P = \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix}. \tag{91}$$

Note that $P\Big(\alpha |0\rangle + \beta |+\rangle\Big) = \alpha |0\rangle$ (for all $\alpha, \beta \in \mathbb{C}$), but $|0\rangle$ and $|+\rangle$ are not orthogonal; this is an *oblique* projector. Throughout these notes, our convention is that *projector* means *orthogonal projector*, unless explicitly stated otherwise.

8.2 Local measurements

The definition of a projective measurement enables us to make sense of scenarios where we measure a *subset* of n qubits. Consider the example where there are two qubits, and we want to measure (only) the first qubit.

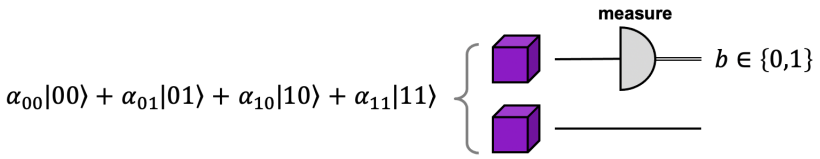

Figure 56: Notation for measuring individual qubits.

The half-circle shape in the circuit diagram is our way of denoting a measurement of an individual qubit. Notice that the wire coming out of the measurement gate is a double line. We can think of the double line as a "thicker wire" that carries classical bits. The outcome of the measurement is either 0 or 1, and the residual state of the first qubit will be either $|0\rangle$ or $|1\rangle$ and the second qubit remains "unmeasured."

Since the original state of the two-qubit system might be entangled, we cannot just ignore the second qubit and use our previous definition for measuring a one-qubit system. There might not be a state vector for the first qubit.

Before we define this measurement in terms of projectors, let's try to visualize the underlying geometry in terms of the following two 2-dimensional subspaces:

- The space of all linear combinations of $|00\rangle$ and $|01\rangle$ (states with first qubit $|0\rangle$).
- The space of all linear combinations of $|10\rangle$ and $|11\rangle$ (states with first qubit $|1\rangle$).

These two spaces are orthogonal to each other (every vector in one space is orthogonal

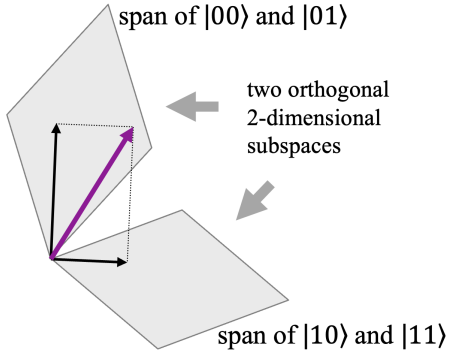

Figure 57: Schematic picture of two orthogonal 2-dimensional spaces in four dimensions.

to every vector in the other space). So we have two orthogonal 2-dimensional spaces within the 4-dimensional space of two-qubit states.

Any two-qubit quantum state $\alpha_{00}|00\rangle+\alpha_{01}|01\rangle+\alpha_{10}|10\rangle+\alpha_{11}|11\rangle$ has a projection onto each subspace. These projections are:

$$\alpha_{00}|00\rangle + \alpha_{01}|01\rangle = |0\rangle \otimes \Big(\alpha_{00}|0\rangle + \alpha_{01}|1\rangle\Big) \tag{92}$$

$$\alpha_{10}|10\rangle + \alpha_{11}|11\rangle = |1\rangle \otimes \Big(\alpha_{10}|0\rangle + \alpha_{11}|1\rangle\Big). \tag{93}$$

And the respective lengths squared of these projections are

$$|\alpha_{00}|^2 + |\alpha_{01}|^2 \tag{94}$$

$$|\alpha_{10}|^2 + |\alpha_{11}|^2. \tag{95}$$

Now we define the outcome of the *measurement of the first qubit* as

$$\begin{cases} 0 \text{ and residual state } \alpha_{00}|0\rangle + \alpha_{01}|1\rangle & \text{with probability } |\alpha_{00}|^2 + |\alpha_{01}|^2 \\ 1 \text{ and residual state } \alpha_{10}|0\rangle + \alpha_{11}|1\rangle & \text{with probability } |\alpha_{10}|^2 + |\alpha_{11}|^2. \end{cases} \tag{96}$$

Note that, in Eq. (96), we are omitting the residual state of the first (measured) qubit, which is $|0\rangle$ or $|1\rangle$, in correspondence with the classical output bit.

The projectors that describe this measurement are

$$\Pi_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \text{and} \quad \Pi_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{97}$$

There is an obvious version of this definition for measuring the *second* qubit of two qubits.

Exercise 8.1 (straightforward)**.** Using a similar approach to the above, propose a definition for the result of measuring the *second* qubit of a 2-qubit system.

This definition of *local measurement* extends in a very straightforward way to the scenario where there are n qubits and some arbitrary subset of k of the qubits are measured. The outcome is a k-bit string, and associated with each outcome, is a 2^{n-k}-dimensional subspace (orthogonal projector of rank 2^{n-k}). There are are 2^k such subspaces (orthogonal to each other) and the outcome probabilities correspond to the projection lengths squared of the state on the 2^k subspaces.

Exercise 8.2 (a straightforward check of the definitions)**.** Consider the 3-qubit state $\frac{1}{\sqrt{2}}|001\rangle + \frac{1}{\sqrt{3}}|010\rangle + \frac{1}{\sqrt{6}}|100\rangle$. What are the outcome probabilities and residual states if the first qubit is measured? What about the case where the second qubit is measured? And if the third qubit is measured?

A state distinguishing problem with local measurements

Recall that the Bell basis is

$$\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle \tag{98}$$

$$\frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle \tag{99}$$

$$\frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle \tag{100}$$

$$\frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|10\rangle\,. \tag{101}$$

Consider the state distinguishing problem where one of these states is created and the goal is to determine which one. This is easy to do perfectly if you're provided with both qubits of the state.

Suppose that we add a restriction that *you are provided with only the first qubit of the state (the second qubit is inaccessible to you).* The trivial baseline strategy for distinguishing among the four Bell states is to randomly guess (without even measuring the qubit), which succeeds with probability $\frac{1}{4}$. It turns out that, if you can only measure the first qubit, then it is impossible to do any better than the baseline strategy.

Exercise 8.3. Suppose that a Bell state is selected randomly and the only information that you are given is the first qubit of the Bell state. Show that there is no measurement (of your qubit) that enables you to guess which Bell state was selected with probability greater than $\frac{1}{4}$.

8.3 Local actions do not non-locally affect measurements

Suppose that Alice and Bob have separate labs, each containing one qubit of the Bell state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$. Now suppose that Alice measures her qubit (in the computational basis). The result will be an unbiased random bit (0 with probability $\frac{1}{2}$ and 1 with probability $\frac{1}{2}$).

How does the situation change if Bob measures his system *before* Alice measures hers? According to our definition of local measurements in section 8.2, if Bob's

measurement outcome is (say) 0 then the state of Alice's qubit becomes $|0\rangle$, *even though Alice has not performed her measurement yet.* Has Bob's measurement of his qubit affected the state of Alice's qubit? On the face of it, one might be tempted to think that it has. Here, I will explain why it's more accurate to view Bob's measurement as having no causal effect on the state of Alice's system.

In the aforementioned scenario, Alice and Bob's individual measurement outcomes are random bits, whose joint distribution is completely correlated. It's as if a fair coin has been flipped and, prior to seeing the outcome, it's a uniformly distributed random bit for both Alice and Bob. If Bob looks at the outcome first then *he* knows what outcome Alice will see. But his looking doesn't affect what Alice will see.

More generally, suppose that Alice and Bob's separate labs each contain one qubit of the state $\alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$. If they measure simultaneously then the joint distribution of the four possible outcomes (00, 01, 10, and 11) is

$$\begin{array}{ll} |\alpha_{00}|^2 & |\alpha_{01}|^2 \\ |\alpha_{10}|^2 & |\alpha_{11}|^2, \end{array} \tag{102}$$

and Alice and Bob's individual measurement outcome bits are the marginals of this distribution: Alice's outcome bits are distributed as the row sums $|\alpha_{00}|^2 + |\alpha_{01}|^2$ and $|\alpha_{10}|^2 + |\alpha_{11}|^2$. Bob's outcome bits are distributed as the column sums $|\alpha_{00}|^2 + |\alpha_{10}|^2$ and $|\alpha_{01}|^2 + |\alpha_{11}|^2$. If one party measures first then it does not affect the marginal distribution of the other party. And this argument extends to higher dimensional systems than pairs of qubits.

In pictures, the local measurements commute with each other.

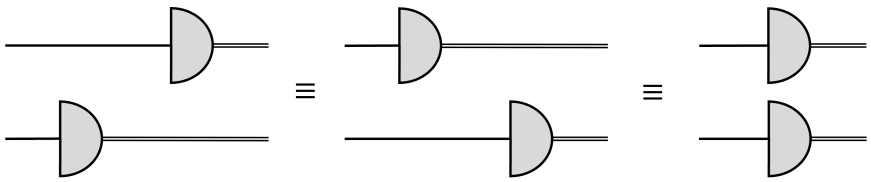

Figure 58: Measuring two separate qubits can be done in any order.

Also, note that if the spacial positions and times of the local measurement are "space-like separated events" (in the language of special relativity) then it is not even well-defined which event occurs first; it depends on the reference frame from which they are being considered. Therefore, the measurement operations need to commute with each other for the theory to make sense. The commutativity property is a sanity-check that our theory passes.

What about if Bob performs a unitary operation on his qubit prior to Alice's measurement? Can this affect the outcome probabilities of a subsequent measurement of Alice? To analyze this, it's convenient to express

$$\begin{aligned}\alpha_{00}\,|00\rangle + \alpha_{01}\,|01\rangle + \alpha_{10}\,|10\rangle + \alpha_{11}\,|11\rangle \\ = \beta_0\,|0\rangle \otimes \Big(\gamma_0\,|0\rangle + \gamma_1\,|1\rangle\Big) + \beta_1\,|1\rangle \otimes \Big(\delta_0\,|0\rangle + \delta_1\,|1\rangle\Big),\end{aligned} \tag{103}$$

where

$$\beta_0 = \sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2} \qquad \beta_1 = \sqrt{|\alpha_{10}|^2 + |\alpha_{11}|^2} \tag{104}$$

$$\gamma_0 = \alpha_{00}/\beta_0 \qquad \gamma_1 = \alpha_{01}/\beta_0 \tag{105}$$

$$\delta_0 = \alpha_{10}/\beta_1 \qquad \delta_1 = \alpha_{11}/\beta_1. \tag{106}$$

From this, it's clear that applying unitary U to the second qubit changes the state to

$$\beta_0\,|0\rangle \otimes U\Big(\gamma_0\,|0\rangle + \gamma_1\,|1\rangle\Big) + \beta_1\,|1\rangle \otimes U\Big(\delta_0\,|0\rangle + \delta_1\,|1\rangle\Big), \tag{107}$$

which has no effect on the outcome probabilities of Alice's subsequent measurement, which are β_0^2 and β_1^2.

Later on, in *Part III: quantum information theory*, we'll learn a language that enables us to express matters like this more clearly.

8.4 Weirdness of the Bell basis encoding

Suppose that we have two qubits which we want to use to encode two *classical* bits. Let's consider two different ways of encode the two classical bits: the computational basis and the Bell basis.

ab	Comp. basis
00	$\|00\rangle$
01	$\|01\rangle$
10	$\|10\rangle$
11	$\|11\rangle$

ab	Bell basis
00	$\|00\rangle + \|11\rangle$
01	$\|00\rangle - \|11\rangle$
10	$\|01\rangle + \|10\rangle$
11	$\|01\rangle - \|10\rangle$

Now, if the two qubits are considered as one system, it doesn't make much difference which encoding you use, because you can always convert between these encodings by a unitary operation. However, if the two qubits are localized: say, Alice possesses the first qubit and Bob possesses the second qubit then there's an interesting difference.

For the case of the computational basis encoding, Alice can determine the value of the first bit a, but not the second bit, b. Also, Alice can flip the value of the first bit (between 0 and 1) but *cannot* flip the second bit. She has complete control over the first bit, but no access to the second bit.

On the other hand, for the case of the Bell basis encoding, Alice has no idea about either bit (she cannot determine any information about the value of a nor the value of b). However, Alice can flip *either one* of the two bits: she can flip the first bit (by applying a Pauli X); she can flip the second bit by applying the Pauli Z); and she can flip both bits, by applying both of these Paulis.

Informally, by using the Bell basis encoding, each party individually forgoes the ability to *read* any of the bits being encoded, but gains the advantage of being able to *flip* both bits by a local operation on just one of the qubits.

This weirdness of the Bell basis is the driving force behind superdense coding (section 7).

8.5 Measuring the control qubit of a controlled-U gate

In section 6.8, we saw that controlled-U gates can behave in remarkable ways, such as changing the state of their control qubit. Here, we consider another phenomenon, which arises when the control-qubit of a controlled-U gate is measured. It is summarized by the equivalence of the following two circuit diagrams.

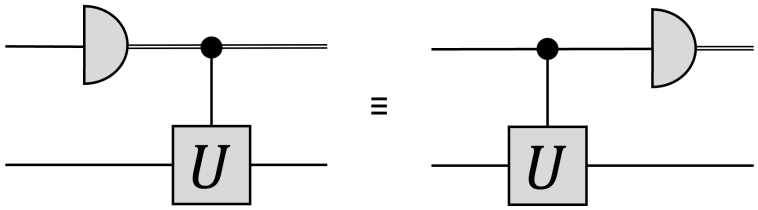

Figure 59: Measuring the control qubit before or after a controlled-U gate.

In the circuit on the left side, the first qubit is measured with respect to the computational basis (yielding outcome 0 or 1) which then serves as the (classical) control of the subsequent controlled-U gate. In the circuit on the right side, the controlled-U gate is performed first (on a fully quantum state) and then the first qubit is measured in the computational basis.

Lemma 8.1 (deferred measurement lemma)**.** *For any 2-qubit input state, show that the effect of the two procedures depicted in figure 59 is exactly the same.*

Exercise 8.4 (fairly straightforward)**.** Prove Lemma 8.1.

⚠ **A word of caution:** the equivalence depicted in figure 59 is valid *if the measurement is with respect to the computational basis.* If the measurement is with respect to a different basis then the equivalence does not hold in general.

9 Isometries and exotic measurements

Although unitary operations are fundamental, there exist more general operations that are clearly within the framework of quantum information processing. For example, suppose that you receive a qubit as *input* and you produce two qubits as *output*: the input qubit (unchanged); plus a second qubit that you yourself set to state $|0\rangle$. Clearly, this is an allowable operation. This mapping $|\psi\rangle \mapsto |\psi\rangle \otimes |0\rangle$ can be mathematically expressed by the matrix

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \tag{108}$$

where multiplying any 2-dimensional state vector $|\psi\rangle$ by the matrix results in the 4-dimensional state vector $|\psi\rangle \otimes |0\rangle$. This is a special case of an *isometry*, defined in the next subsection.

In subsequent subsections, we will see how we can use isometries to extend our repertoire of measurements in interesting ways.

9.1 Isometries

Definition 9.1. An *isometry* is an $m \times n$ matrix V such that $V^*V = I$.

Note that this only makes sense when $m \geq n$, since otherwise $V^*V = I$ cannot hold. In the special case where $m = n$, V is unitary; in the more general case, V is characterized by the property that its columns are orthonormal.

Exercise 9.1 (isometries preserve inner products)**.** Show that an $m \times n$ matrix V is an isometry if and only if, for all n-dimensional states $|\psi\rangle$ and $|\phi\rangle$, the inner product between $V|\psi\rangle$ and $V|\phi\rangle$ is the same as the inner product between $|\psi\rangle$ and $|\phi\rangle$.

A very simple example of an isometry is

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, \tag{109}$$

which maps any qubit in state $\alpha_0|0\rangle + \alpha_1|1\rangle$ to a qutrit in state $\alpha_0|0\rangle + \alpha_1|1\rangle$. This embeds a 2-dimensional state into the first two dimensions of a 3-dimensional space.

Another, less trivial, example of an isometry is

$$\begin{bmatrix} \frac{2}{\sqrt{6}} & 0 \\ -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} \end{bmatrix}. \tag{110}$$

We'll see an application of this partricular isometry later on.

9.2 Exotic measurements

Now is a good time to see the *exotic measurements* that I briefly alluded to at the end of section 3.4, but have not yet explained.

First, to review some measurements, we have our basic measurement operation, which is with respect to the computational basis states. We also have a notion of measuring a qubit with respect to any orthonormal basis, which can be simulated by preceding a basic measurement with some unitary operation U. The more exotic measurements precede the measurement in the computational basis by an isometry.

If you are hearing about this kind of measurement process for the first time, you might wonder what the point of doing this is. Is there anything special that measuring with respect to an orthonormal basis in a higher-dimensional space can achieve? In fact this can be very useful. In subsections 9.2.1 and 9.2.2, I'll show you some interesting examples.

9.2.1 Trine state distinguishing

Recall the trine state measurement problem that arose within section 5.1.2, where the goal is to distinguish among these three one-qubit states (with equal angles between them):

$$|\phi_0\rangle = |0\rangle \tag{111}$$

$$|\phi_1\rangle = -\tfrac{1}{2}|0\rangle + \tfrac{\sqrt{3}}{2}|1\rangle \tag{112}$$

$$|\phi_2\rangle = -\tfrac{1}{2}|0\rangle - \tfrac{\sqrt{3}}{2}|1\rangle. \tag{113}$$

In the proof of Theorem 5.2, we saw how to use qubit measurements to correctly determine the state with probability $\frac{7}{12}$. Also, it was mentioned that this can be improved to $\frac{2}{3}\cos^2(\frac{\pi}{12}) \approx 0.622$ (left as Exercise 5.1). Here, we show how to use an exotic measurement to further improve the success probability to $\frac{2}{3}$.

The idea is based on a nice geometric arrangement of vectors in three dimensions, described in figures 60 and 61.

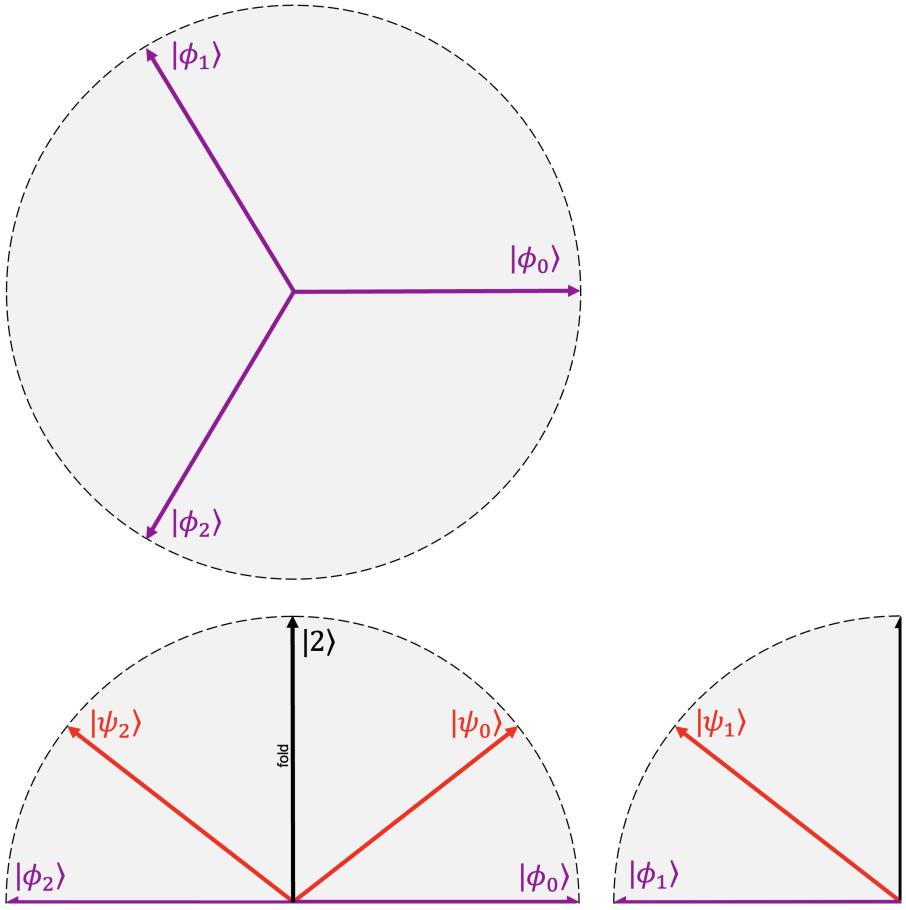

Figure 60: Visualization of trine state measurement: cut out these templates (or imagine doing so).

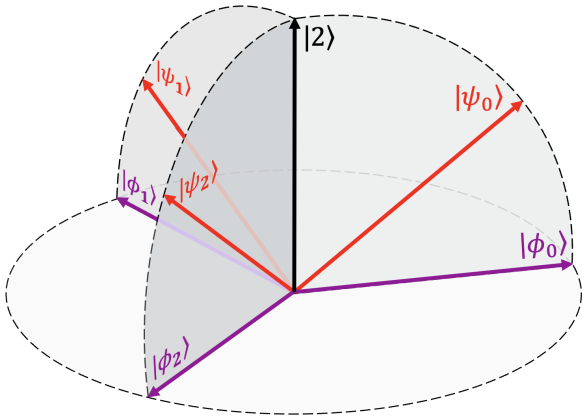

Figure 61: Visualization of trine state measurement: assemble the cutouts in 3-dimensions like this.

The angle of the red vectors is set so that, in the 3-dimensional arrangement of figure 61, they are mutually orthogonal. This is achieved by setting

$$|\psi_k\rangle = \sqrt{\tfrac{2}{3}}\,|\phi_k\rangle + \tfrac{1}{\sqrt{3}}\,|2\rangle\,, \tag{114}$$

for $k \in \{0, 1, 2\}$. It is straightforward to check that $|\psi_0\rangle$, $|\psi_1\rangle$, $|\psi_2\rangle$ are orthogonal:

$$\langle\psi_j|\psi_k\rangle = \tfrac{2}{3}\,\langle\phi_j|\phi_k\rangle + \tfrac{1}{3}\,\langle 2|2\rangle = \tfrac{2}{3}(-\tfrac{1}{2}) + \tfrac{1}{3} = 0, \tag{115}$$

for all $j \neq k$.

Our distinguishing procedure is to first map our 2-dimensional input state into a 3-dimensional space by performing the isometry

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{116}$$

and then measure with respect to the orthonormal basis $|\psi_0\rangle$, $|\psi_1\rangle$, $|\psi_2\rangle$. When the input state is $|\phi_k\rangle$, the probability that the outcome is k is $|\langle\psi_k|\phi_k\rangle|^2 = \frac{2}{3}$. Therefore, this measurement correctly identifies among the states $|\phi_0\rangle$, $|\phi_1\rangle$, $|\phi_2\rangle$ with success probability $\frac{2}{3}$.

This procedure can be shown to be equivalent to performing the isometry

$$\begin{bmatrix} \frac{2}{\sqrt{6}} & 0 \\ -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \tag{117}$$

and then measuring with respect to the 3-dimensional computational basis.

A variant of the trine state measurement problem

Suppose that you receive (one copy) of a trine state that you are allowed to measure, and after your measurement you are allowed to make *two* guesses at the state. Your answer is deemed successful if one of your guesses is correct.

If we use the above measurement, then it can be worked out that the success probability for this two-guess game will be $\frac{2}{3} + \frac{1}{2} = \frac{5}{6}$. But there is a different measurement that leads to success probability 1 for this two-guess game.

Exercise 9.2. Give a different measurement procedure for a trine state, whose outcome enables you to produce two guesses such that one of them is guaranteed to be correct. (Hint: If you are stuck with this then you might consider playing around with the cutouts from figures 60 and 61.)

9.2.2 Zero-error state distinguishing between $|0\rangle$ and $|+\rangle$

Here we consider a variant of the problem of distinguishing between the two states $|0\rangle$ and $|+\rangle$. In section 4.1, we saw how to determine the state correctly with success probability $\cos^2(\pi/8) = 0.853...$, but no higher. Note that this procedure gives the wrong answer with probability $\sin^2(\pi/8) = 0.146...$.

A *zero-error* procedure for state distinguishing *never* gives the wrong answer.[9] But that does not mean it always gives the right answer. This is because the procedure is allowed to sometimes *abstain* from giving an answer. Formally, the potential outputs of the distinguishing procedure are $\{0, 1, A\}$, where:

- 0 means a guess that the state is $|0\rangle$.
- 1 means a guess that the state is $|+\rangle$.
- A means "abstain" (in other words, no guess).

To be *zero-error* means that an output of 0 or 1 is always correct.

Now there's a very trivial zero-error procedure: abstain all the time. But that's not interesting, because it never guesses the state correctly either. A nontrivial zero-error procedure is one that *sometimes* does not abstain (and in such cases, the guess has to be correct).

If we have a zero-error-procedure, it's *success probability* on an input instance is defined as the probability that it gives the right answer for that input.

Imagine a situation where you can make a guess about something. When you are right you are rewarded; when you are wrong you are penalized. But you also have the option of abstaining, in which case you get no reward and no penalty. Maybe the penalty for a wrong guess is extremely high so you cannot afford to ever make a wrong guess. But you'd still like to *sometimes* get the reward, so you don't want to always abstain.

What is the best zero-error success probability for distinguishing between $|0\rangle$ and $|+\rangle$? The idea that follows is based on a nice geometric arrangement of vectors in three dimensions. To see it, you can cut out the rectangle in figure 62 and fold it 90 degrees in the middle (or imagine doing so).

[9] This is sometimes called *unambiguous state discrimination*.

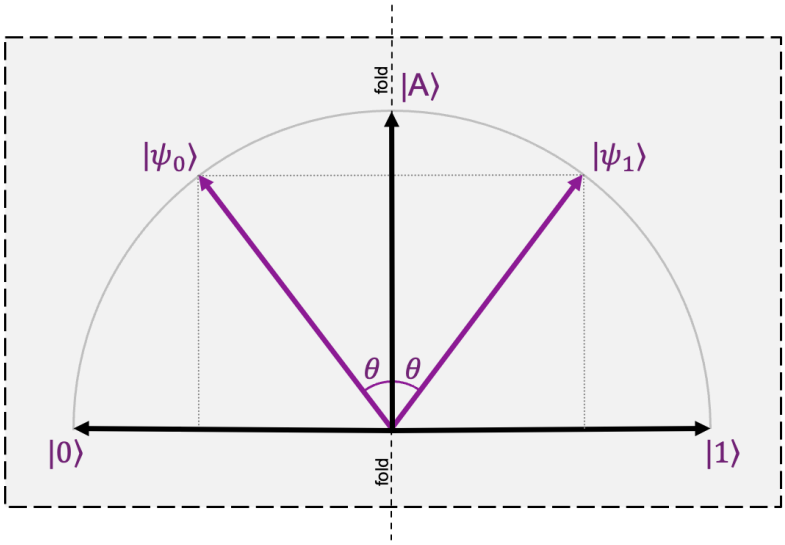

Figure 62: Template for a special geometric arrangement of vectors (fold 90 degrees in the middle).

The result will look like figure 63.

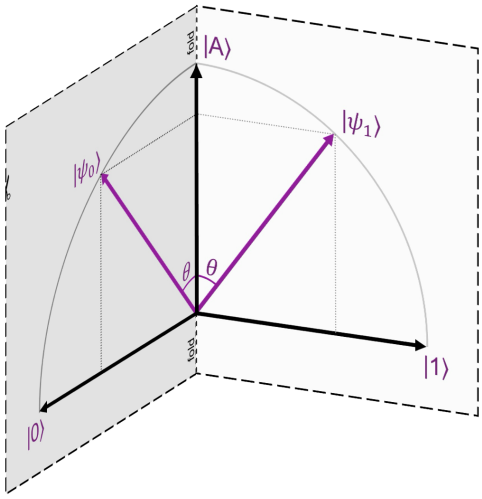

Figure 63: Special geometric arrangement of vectors.

Note that the states $|0\rangle$, $|1\rangle$, and $|A\rangle$ are mutually orthogonal and the states $|\psi_0\rangle$ and $|\psi_1\rangle$ are of the form

$$|\psi_0\rangle = \sin(\theta)\,|0\rangle + \cos(\theta)\,|A\rangle \tag{118}$$

$$|\psi_1\rangle = \sin(\theta)\,|1\rangle + \cos(\theta)\,|A\rangle\,. \tag{119}$$

We set the angle θ so that $\langle\psi_0|\psi_1\rangle = \langle 0|+\rangle = \frac{1}{\sqrt{2}}$, which occurs when $\cos^2(\theta) = \frac{1}{\sqrt{2}}$. Since the inner product between $|\psi_0\rangle$ and $|\psi_1\rangle$ equals the inner product between $|0\rangle$

and $|+\rangle$, there exists an isometry that maps $|0\rangle$ to $|\psi_0\rangle$ and $|+\rangle$ to $|\psi_1\rangle$. For now, I'm appealing to your geometric intuition that the two-dimensional space of qubits can be embedded into three dimensions in this manner; however, I will later make this isometry explicit.

After performing the isometry, our goal is to distinguish between $|\psi_0\rangle$ and $|\psi_1\rangle$ with zero-error. We do this by measuring with respect to the basis $|0\rangle$, $|1\rangle$, and $|A\rangle$. The outcome probabilities for the two cases, $|\psi_0\rangle$ and $|\psi_1\rangle$, can be deduced from equations (118) and (119) as follows. For state $|\psi_0\rangle$, the outcome probabilities are

$$\begin{cases} 0 & \text{with probability } \sin^2(\theta) \\ 1 & \text{with probability } 0 \\ A & \text{with probability } \cos^2(\theta), \end{cases} \tag{120}$$

and, for state $|\psi_1\rangle$, the outcome probabilities are

$$\begin{cases} 0 & \text{with probability } 0 \\ 1 & \text{with probability } \sin^2(\theta) \\ A & \text{with probability } \cos^2(\theta). \end{cases} \tag{121}$$

It follows that this procedure is zero-error and the success probability in each case is $\sin^2(\theta) = 1 - \frac{1}{\sqrt{2}} = 0.292...$ (since we have set θ so that $\cos^2(\theta) = \frac{1}{\sqrt{2}}$).

I have shown that there exists an exotic measurement that performs the zero-error state distinguishing with success probability $1 - \frac{1}{\sqrt{2}}$, but without explicitly specifying the isometry that maps $|0\rangle$ and $|+\rangle$ (respectively) to $|\psi_0\rangle$ and $|\psi_1\rangle$ in figure 63. I will now give an explicit isometry.

It's convenient to start with a preliminary rotation by angle $\pi/8$ to the input state (which is $|0\rangle$ or $|+\rangle$) so that the states to distinguish become

$$|\phi_0\rangle = \cos(\pi/8)\,|0\rangle + \sin(\pi/8)\,|1\rangle \tag{122}$$

$$|\phi_1\rangle = \sin(\pi/8)\,|0\rangle + \cos(\pi/8)\,|1\rangle\,. \tag{123}$$

For the states in this form, we can use the isometry

$$V = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} - 1 \\ \frac{1}{\sqrt{2}} - 1 & \frac{1}{\sqrt{2}} \\ \sqrt{\sqrt{2} - 1} & \sqrt{\sqrt{2} - 1} \end{bmatrix}. \tag{124}$$

How do we know that this works? First of all, we can verify that V is an isometry, by checking that its columns are orthonormal. And it is possible determine that V results in zero-error success probability $1 - \frac{1}{\sqrt{2}}$ by calculating that

$$V|\phi_0\rangle = \sqrt{1 - \tfrac{1}{\sqrt{2}}}\,|0\rangle + \sqrt{\tfrac{1}{\sqrt{2}}}\,|A\rangle \tag{125}$$

$$V|\phi_1\rangle = \sqrt{1 - \tfrac{1}{\sqrt{2}}}\,|1\rangle + \sqrt{\tfrac{1}{\sqrt{2}}}\,|A\rangle\,, \tag{126}$$

where the calculation makes use of the fact[10] that

$$\cos(\pi/8) = \tfrac{1}{\sqrt{2}}\sqrt{1 + \tfrac{1}{\sqrt{2}}} \tag{127}$$

$$\sin(\pi/8) = \tfrac{1}{\sqrt{2}}\sqrt{1 - \tfrac{1}{\sqrt{2}}}. \tag{128}$$

[10] Which follows from $\cos^2(\pi/8) = \frac{1+\cos(\pi/4)}{2}$ and $\sin^2(\pi/8) = \frac{1-\cos(\pi/4)}{2}$.

10 Teleportation

Consider the problem where Alice wants to communicate an arbitrary qubit to Bob by sending only *a finite number of classical bits*. Intuitively, one might expect that, since there are a continuum of possible qubit state vectors, this is impossible to accomplish. Teleportation[11] violates this intuition, though it makes use of an extra resource: entanglement between Alice and Bob.

10.1 Prelude to teleportation

Consider the scenario where Alice receives a qubit as input and the goal is for her to convey it to Bob. Based on the qubit that Alice receives, she determines some classical bits to send to Bob. When Bob receives these classical bits, he is supposed to reconstruct Alice's original state.

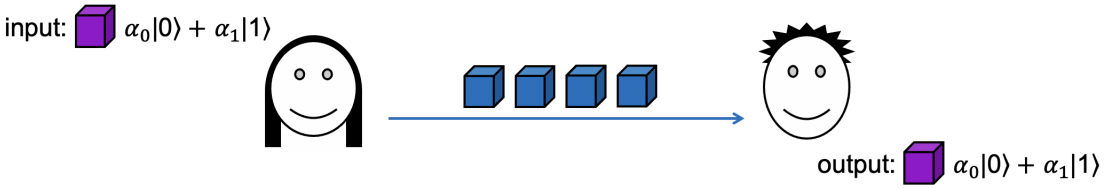

Figure 64: Communicating a qubit by sending classical bits.

If Alice *knows* the state $\alpha_0 |0\rangle + \alpha_1 |1\rangle$ of the qubit she receives then she can send bits that specify α_0 and α_1 within some precision. High precision would require Alice sending many bits—and perfect precision would require infinitely many bits. Moreover, the situation is even worse than that: Alice might not even *know* the amplitudes of the qubit that she received. Maybe the state was set by a third party, who gave the qubit to Alice without telling her what the state is. Alice can at best obtain one bit of information about the state by measuring it, and that process destroys the state.

10.2 Teleportation scenario

In the teleportation scenario, Alice and Bob start with an additional resource, a shared Bell state $\frac{1}{\sqrt{2}} |00\rangle + \frac{1}{\sqrt{2}} |11\rangle$ and Alice sends Bob only two classical bits.

[11] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters (1993). "Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels." *Phys. Rev. Lett.*, **70**(13): 1895–1899.

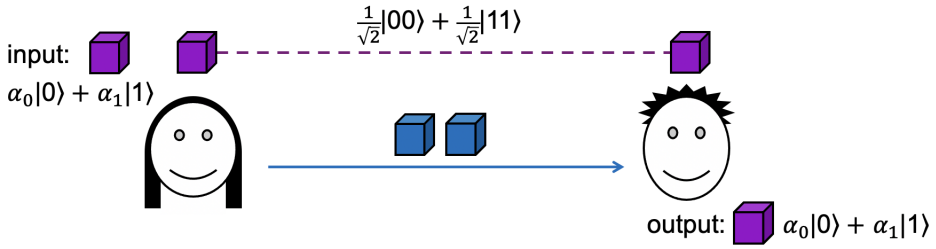

Figure 65: Teleportation scenario.

Note that the Bell state contains absolutely no information about Alice's input state $\alpha_0 |0\rangle + \alpha_1 |1\rangle$. It is remarkable that, in this scenario, there is a protocol where Alice sends two classical bits to Bob and he is able to perfectly reconstruct the state.

10.3 How teleportation works

We begin by considering the initial state of the system, where Alice is in possession of her input qubit and the first qubit of the Bell state and Bob is in possession of the second qubit of the Bell state. We can write this state as

$$\Big(\alpha_0 |0\rangle + \alpha_1 |1\rangle\Big) \otimes \Big(\tfrac{1}{\sqrt{2}} |00\rangle + \tfrac{1}{\sqrt{2}} |11\rangle\Big) \tag{129}$$

$$= \tfrac{1}{\sqrt{2}}\alpha_0 |000\rangle + \tfrac{1}{\sqrt{2}}\alpha_0 |011\rangle + \tfrac{1}{\sqrt{2}}\alpha_1 |100\rangle + \tfrac{1}{\sqrt{2}}\alpha_1 |111\rangle . \tag{130}$$

It is clear that all the information about the state $\alpha_0 |0\rangle + \alpha_1 |1\rangle$ resides with Alice. It is interesting that we can write the state in Eq. (130) as

$$\begin{aligned}
&\tfrac{1}{\sqrt{2}}\alpha_0 |000\rangle + \tfrac{1}{\sqrt{2}}\alpha_0 |011\rangle + \tfrac{1}{\sqrt{2}}\alpha_1 |100\rangle + \tfrac{1}{\sqrt{2}}\alpha_1 |111\rangle \\
&\quad = \tfrac{1}{2}\Big(\tfrac{1}{\sqrt{2}} |00\rangle + \tfrac{1}{\sqrt{2}} |11\rangle\Big) \otimes \Big(\alpha_0 |0\rangle + \alpha_1 |1\rangle\Big) \\
&\qquad + \tfrac{1}{2}\Big(\tfrac{1}{\sqrt{2}} |01\rangle + \tfrac{1}{\sqrt{2}} |10\rangle\Big) \otimes \Big(\alpha_0 |1\rangle + \alpha_1 |0\rangle\Big) \\
&\qquad + \tfrac{1}{2}\Big(\tfrac{1}{\sqrt{2}} |00\rangle - \tfrac{1}{\sqrt{2}} |11\rangle\Big) \otimes \Big(\alpha_0 |0\rangle - \alpha_1 |1\rangle\Big) \\
&\qquad + \tfrac{1}{2}\Big(\tfrac{1}{\sqrt{2}} |01\rangle - \tfrac{1}{\sqrt{2}} |10\rangle\Big) \otimes \Big(\alpha_0 |1\rangle - \alpha_1 |0\rangle\Big).
\end{aligned} \tag{131}$$

First, it is worth confirming that Eq. (131) is correct.

Exercise 10.1 (straightforward)**.** Confirm Eq. (131). (Hint: expand the tensor products and observe that some of the terms cancel out.)

What is remarkable about the expression in Eq. (131) is that the coefficients α_0 and α_1 appear to be on Bob's side—and the teleportation protocol has not even started! How did α_0 and α_1 migrate over to Bob's side? In spite of Eq. (131), Bob's qubit contains absolutely no information about $\alpha_0 |0\rangle + \alpha_1 |1\rangle$. We have to be careful not to misinterpret the state in Eq. (131).

But Eq. (131) suggests an approach to make the teleportation protocol work: what if Alice measures her qubits (the first two qubits) *in the Bell basis*? Then, for each outcome, the residual state of Bob's qubit is similar to $\alpha_0 |0\rangle + \alpha_1 |1\rangle$. A simple correction, based on Alice's outcome, can make the state exactly $\alpha_0 |0\rangle + \alpha_1 |1\rangle$.

The measurement with respect to the Bell basis can be accomplished by Alice first applying the 2-qubit unitary operation specified by this circuit.[12]

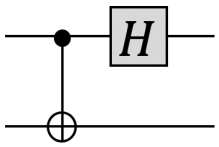

Figure 66: Circuit that converts from the Bell basis to the computational basis.

This has the following effect on the Bell states

input	output
$\|00\rangle + \|11\rangle$	$\|00\rangle$
$\|00\rangle - \|11\rangle$	$\|01\rangle$
$\|01\rangle + \|10\rangle$	$\|10\rangle$
$\|01\rangle - \|10\rangle$	$\|11\rangle$

Therefore this changes the 3-qubit state to

$$\begin{aligned} &\tfrac{1}{2} |00\rangle \otimes \Big(\alpha_0 |0\rangle + \alpha_1 |1\rangle\Big) \\ +\, &\tfrac{1}{2} |01\rangle \otimes \Big(\alpha_0 |1\rangle + \alpha_1 |0\rangle\Big) \\ +\, &\tfrac{1}{2} |10\rangle \otimes \Big(\alpha_0 |0\rangle - \alpha_1 |1\rangle\Big) \\ +\, &\tfrac{1}{2} |11\rangle \otimes \Big(\alpha_0 |1\rangle - \alpha_1 |0\rangle\Big). \end{aligned} \tag{132}$$

Now, if Alice measures the first two qubits of this state in the computational basis

[12] We also used this circuit for superdense coding (section 7.2, figure 54).

then the result (Alice's two classical bits and the residual state in Bob's possession) is

$$\begin{cases} 00, & \alpha_0 |0\rangle + \alpha_1 |1\rangle & \text{with probability } \frac{1}{4} \\ 01, & \alpha_0 |1\rangle + \alpha_1 |0\rangle & \text{with probability } \frac{1}{4} \\ 10, & \alpha_0 |0\rangle - \alpha_1 |1\rangle & \text{with probability } \frac{1}{4} \\ 11, & \alpha_0 |1\rangle - \alpha_1 |0\rangle & \text{with probability } \frac{1}{4}. \end{cases} \tag{133}$$

At this point, Bob does not yet have the correct state (expect in the case of outcome 00). But, if Alice sends Bob the two bits of her measurement outcome then Bob can apply an appropriate operation to "correct" his state.

Here's what Bob does after receiving the two classical bits ab from Alice:

1. If $b = 1$ apply X.
2. If $a = 1$ apply Z.

The resulting state on Bob's side is for each case is

$$\begin{cases} 00, & \alpha_0 |0\rangle + \alpha_1 |1\rangle \\ 01, & X\Big(\alpha_0 |1\rangle + \alpha_1 |0\rangle\Big) = \alpha_0 |0\rangle + \alpha_1 |1\rangle \\ 10, & Z\Big(\alpha_0 |0\rangle - \alpha_1 |1\rangle\Big) = \alpha_0 |0\rangle + \alpha_1 |1\rangle \\ 11, & ZX\Big(\alpha_0 |1\rangle - \alpha_1 |0\rangle\Big) = \alpha_0 |0\rangle + \alpha_1 |1\rangle. \end{cases} \tag{134}$$

This completes the description of the teleportation protocol.

The protocol can be summarized by the following circuit. Alice's qubits and bits

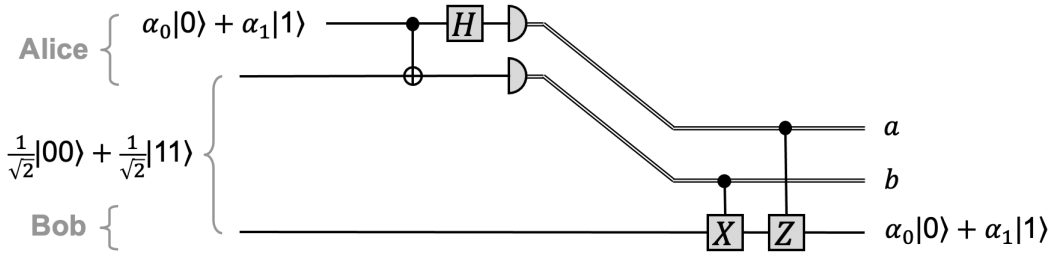

Figure 67: The teleportation protocol summarized in one circuit.

are on top and Bob's are on the bottom. The slanted classical wires denote that the two classical bits resulting from Alice's measurements being shifted down from Alice towards Bob.

Exercise 10.2 (straightforward)**.** Work through the circuit diagram in figure 67 and confirm that it works.

It is natural to ask: What happens to Alice's copy of her state? Is Alice's copy preserved? The answer is that, since Alice measures her two qubits, all the quantum information in her possession is lost. So, while Bob ends up with a copy of the state $\alpha_0 |0\rangle + \alpha_1 |1\rangle$, Alice loses her copy of the state in the teleportation process.

11 Can quantum states be copied?

In the teleportation protocol, Alice loses her copy while Bob obtains a copy. Can this protocol be modified so that Alice's copy is not lost? Or is there some other way to produce a second copy of a quantum state?

11.1 A classical bit copier

Classical information is easy to copy and we do it all the time (say, when we back up our data).

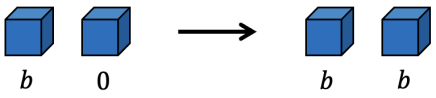

Figure 68: Classical *bit* copying (where $b \in \{0, 1\}$).

A simple device that copies one bit could be conceived as a gate of this form.

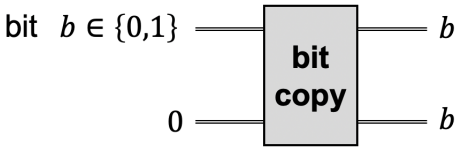

Figure 69: A circuit diagram for a bit copying device.

The first input bit is the data to be copied. The second input bit is always 0 (think of it as analogous to the blank sheet of paper that goes into a photocopier).

How do we implement such a device? It is not hard to see that a CNOT gate (a classical version of this gate) will perform the copying operation.

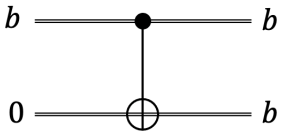

Figure 70: A classical version of the CNOT gate is a bit copier (where $b \in \{0, 1\}$).

11.2 A qubit copier?

A *qubit* copier would perform the following transformation.

Figure 71: A hypothetical qubit copier.

We can try to conceive of such a device as a gate of the following form.

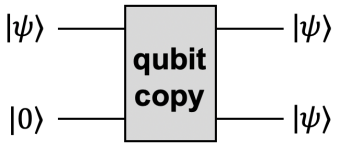

Figure 72: Form of a hypothetical *qubit copier*.

Does there exist a unitary operation that performs this for any input state $|\psi\rangle$?

Our first candidate might be the quantum CNOT gate. Does this work?

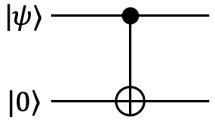

Figure 73: A candidate for a qubit copier.

The CNOT gate actually works correctly for the input states $|0\rangle$ and $|1\rangle$.

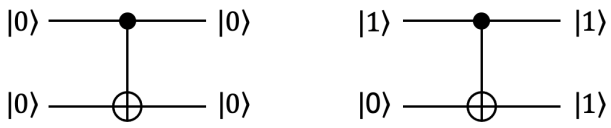

Figure 74: CNOT copies the computational basis states correctly.

However, the CNOT gate fails to correctly copy the state $|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$.

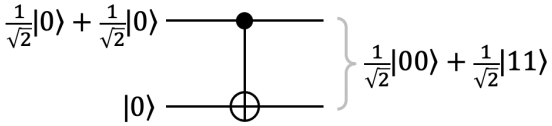

Figure 75: CNOT fails to copy the $|+\rangle$ state.

The output of the gate is $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$, which is not the correct output because two copies of the $|+\rangle$ state is the state $|+\rangle \otimes |+\rangle = \frac{1}{2}|00\rangle + \frac{1}{2}|01\rangle + \frac{1}{2}|10\rangle + \frac{1}{2}|11\rangle$.

Moreover, the No-cloning Theorem[13] implies that it is *impossible* to implement the gate in figure 72.

Theorem 11.1. *There does not exist a 2-qubit unitary that implements the quantum copier in figure 72.*

Proof. If unitary U correctly copies for the states $|0\rangle$ and $|1\rangle$ then

$$U|00\rangle = |00\rangle \tag{135}$$
$$U|10\rangle = |11\rangle\,. \tag{136}$$

It follows by linearity that

$$U|+\rangle|0\rangle = \tfrac{1}{\sqrt{2}}|00\rangle + \tfrac{1}{\sqrt{2}}|11\rangle \neq |+\rangle|+\rangle\,, \tag{137}$$

which means that U does not correctly copy for the state $|+\rangle$. ■

Theorem 11.1 doesn't quite settle the matter of whether quantum information can be copied, because figure 72 is not the most general possible form that a hypothetical qubit copier can take. A more general form is the following. Think of the first qubit

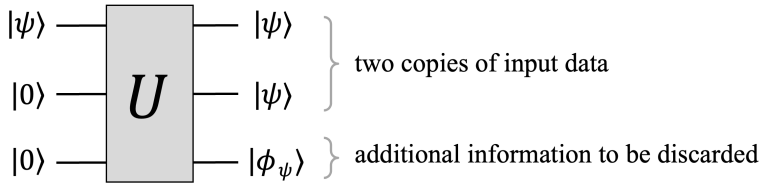

Figure 76: A more general form of a hypothetical quantum copier.

as the data to be copied, the second qubit as the analogue of the blank sheet of paper that goes into a photocopier, and the third register as the analogue of the toner cartridge, which is discarded at the end of the process. The notation for the output state of the third register $|\phi_\psi\rangle$ is intended to indicate that it is allowed to be a function of the data $|\psi\rangle$. In fact, this more general framework does not help.

Theorem 11.2. *There does not exist a 2-qubit unitary that implements the quantum copier in figure 76.*

Exercise 11.1. Prove Theorem 11.2.

[13] William K. Wootters and Wojciech H. Zurek (1982). "A single quantum cannot be cloned." *Nature*, **299**(5886): 802–803.

For completeness, we end this section by stating the general *No-cloning Theorem* for d-dimensional quantum states.

Theorem 11.3 (No-cloning Theorem for d-dimensional states)**.** *For arbitrary dimension $d \geq 2$, there does not exist an ancilla dimension $d' \geq 2$ and a unitary operation U that acts on two d-ary quantum states and one d'-ary state such that, for all $|\psi\rangle \in \mathbb{C}^d$,*

$$U |\psi\rangle |0\rangle |0\rangle = U |\psi\rangle |\psi\rangle |\phi_\psi\rangle , \tag{138}$$

where $|\phi_\psi\rangle$ can be an arbitrary state that depends on $|\psi\rangle$.

12 Can quantum states be stored redundantly?

The No-cloning Theorem (Theorem 11.3) suggests that there can be no redundancy in quantum states, in the sense that it is not possible to construct a "back-up copy" of an arbitrary quantum state. If it *were* possible to construct such a copy then the data would be recoverable if at most one of the two versions were lost.

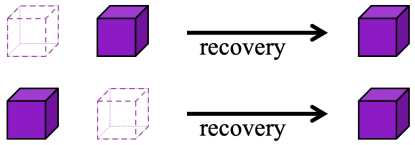

Figure 77: A *hypothetical* duplicate copy enables data to be recovered from the loss of one copy.

However, more subtle forms of redundancy *are* possible. Here, I will show how to do "3/2-copying," which can be thought of as producing three "half-copies" of an

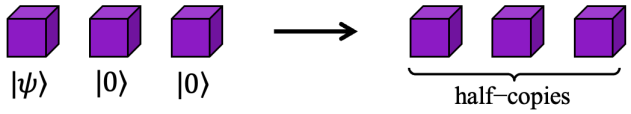

Figure 78: 3/2-copier (encoder part).

arbitrary quantum state, so that the state can be reconstructed from *any two* of the three half-copies. This is redundancy in the sense that the data can be recovered if any one of the three half-copies is lost.

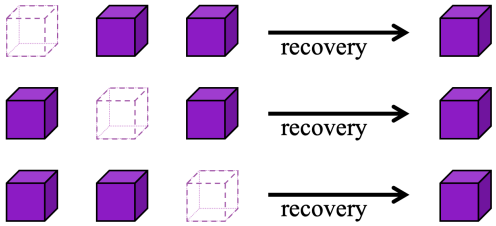

Figure 79: Data can be recovered from any two of the three "half-copies."

I will explain how to do this for the case of *qutrits*, which turns out to be fairly simple. By the way, "3/2-copying" is playful language. The common terminology for constructions along these lines is *erasure codes*, and the construction in the next subsection comes from *quantum secret sharing*.[14]

[14] R. Cleve, D. Gottesman, H.-K. Lo (1999). "How to share a quantum secret." *Phys. Rev. Lett.*, **83**(3): 648–651. arXiv:9901025

12.1 ³⁄₂-copying (via secret sharing)

Here we show how to "encode" a qutrit into three qutrits in a manner where the qutrit can be recovered from *any two* of the three qutrits of the encoding.

An arbitrary qutrit in state $\alpha_0 |0\rangle + \alpha_1 |1\rangle + \alpha_2 |2\rangle$ is *encoded* in three qutrits as the state

$$\begin{aligned} &\alpha_0\Big(|000\rangle + |111\rangle + |222\rangle\Big) \\ +\,&\alpha_1\Big(|012\rangle + |120\rangle + |201\rangle\Big) \\ +\,&\alpha_2\Big(|021\rangle + |102\rangle + |210\rangle\Big). \end{aligned} \tag{139}$$

How can we implement such an encoding in terms of valid quantum operations, such as unitary operations? It can be done, but I will show you how later on.

For now, note the symmetry of the state in Eq. (139) under cyclic shifts. A *cyclic shift by one* is the rearrangement of qutrits where the first qutrit is moved to the position of the second qutrit; the second qutrit is moved to the position of the third qutrit; and the third qutrit is moved to the position of the first qutrit. A *cyclic shift by two* is a composition of two cyclic shifts by one. Figure 80 is circuit notation for such cyclic shifts. You can check by inspection that the state in Eq. (139) does not

Figure 80: Circuits diagrams for cyclic shift by one (left) and cyclic shift by two (right).

change when such cyclic shifts are performed.

To describe the recovery circuits (which enable the original qutrit state to be recovered from any two of the three qutrits) we first introduce a qutrit analogue of the CNOT gate, which we call the ADD_3 gate, which maps $|a\rangle |b\rangle$ to $|a\rangle |b + a \bmod 3\rangle$ (for all $a, b \in \{0, 1, 2\}$). This gate is denoted on qutrit circuits like the CNOT gate.

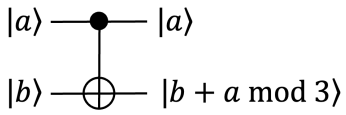

Figure 81: Circuit notation for the ADD_3 gate, a qutrit analogue of the CNOT gate.

Now suppose that we are just given the *first two qutrits* of the encoded state in Eq. (139) (i.e., the third qutrit is lost). The circuit in figure 82 recovers the original

data in the first qutrit. The validity of this circuit follows from setting the input state

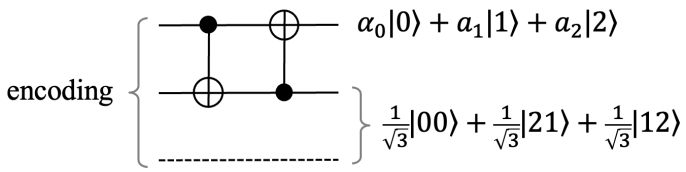

Figure 82: Circuit that recovers the original qutrit from just the first two "half-copies".

to the *encoding* of Eq. (139) and then tracing through the execution of the circuit. Let's do that. After the first ADD_3 gate, the state becomes

$$\begin{aligned} &\alpha_0\Big(|000\rangle + |121\rangle + |212\rangle\Big) \\ +\,&\alpha_1\Big(|012\rangle + |100\rangle + |221\rangle\Big) \\ +\,&\alpha_2\Big(|021\rangle + |112\rangle + |200\rangle\Big), \end{aligned} \tag{140}$$

and after the second ADD_3 gate, the state becomes

$$\begin{aligned} &\alpha_0\Big(\,|000\rangle + |021\rangle + |012\rangle\Big) \\ +\,&\alpha_1\Big(\,|112\rangle + |100\rangle + |121\rangle\Big) \\ +\,&\alpha_2\Big(\,|221\rangle + |212\rangle + |200\rangle\Big) \end{aligned} \tag{141}$$

$$= \Big(\alpha_0\,|0\rangle + \alpha_1\,|1\rangle + \alpha_2\,|2\rangle\Big) \otimes \Big(|00\rangle + |12\rangle + |21\rangle\Big). \tag{142}$$

Also, notice that the two gates of the circuit *only act on the first two qutrits*; the third qutrit isn't needed (the third wire in figure 82 is a dashed-line to highlight this).

Now, consider the case where the *second* qutrit is the lost one. Since the cyclic shift circuits in figure 80 do not change the encoded state, we can imagine applying the cyclic shift by one prior to decoding, which has the effect of moving the position of the missing qutrit from the second to the third position, as shown in figure 83. This enables the data to be recovered from the first and third qutrit.

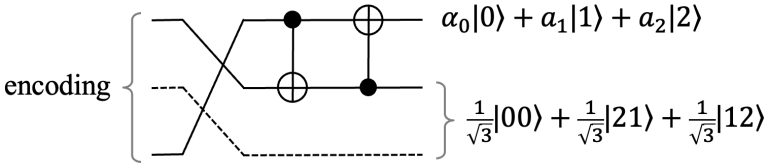

Figure 83: Circuit that recovers the original qutrit from just the first and third "half-copies".

The case where the *first* qutrit is the missing one is similar, where we apply a cyclic shift by two.

Finally, we address how the encoding procedure can be implemented. First note that the *inverse* of the recovery circuit in figure 82 is almost the encoding circuit, as it maps $\left(\alpha_0 |0\rangle + \alpha_1 |1\rangle + \alpha_2 |2\rangle\right) \otimes \left(\frac{1}{\sqrt{3}} |00\rangle + \frac{1}{\sqrt{3}} |12\rangle + \frac{1}{\sqrt{3}} |21\rangle\right)$ to the state of Eq. (139). We need only add some gates to the beginning of the inverse of the circuit of figure 82 that map the two ancilla qutrits from state $|00\rangle$ to state $\frac{1}{\sqrt{3}} |00\rangle + \frac{1}{\sqrt{3}} |12\rangle + \frac{1}{\sqrt{3}} |21\rangle$. A circuit along these lines is in figure 84, where the gates are defined below.

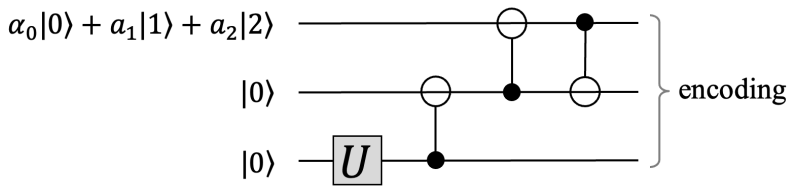

Figure 84: Circuit that produces the encoding state of Eq. (139).

The gate U is a 3×3 unitary operation that maps $|0\rangle$ to $\frac{1}{\sqrt{3}} |0\rangle + \frac{1}{\sqrt{3}} |1\rangle + \frac{1}{\sqrt{3}} |2\rangle$. An example of such a U is

$$U = \begin{bmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{3}} & -\frac{2}{\sqrt{6}} & 0 \end{bmatrix} \tag{143}$$

(another example is the Fourier transform modulo 3, defined in section 19, Eq. (182)).

The other gate appearing in figure 84 is not the ADD_3 gate (notice that it looks slightly different). Rather, it is the *inverse* of the ADD_3 gate, defined as follows.

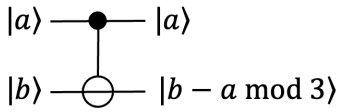

Figure 85: Circuit notation for the inverse of the ADD_3 gate.

This completes the description of ⅔-copying a qutrit. What about states of dimensions other than 3? A construction along these lines exists for certain higher dimensions, such as prime numbers larger than 3; however, it does not work directly for dimension 2 in the sense that a qubit cannot be ⅔-copied into three qubits.[15]

[15] M. Grassl, Th. Beth, and T. Pellizzari (1997). "Codes for the quantum erasure channel." *Phys. Rev. A*, **56**(1): 33–38. arXiv:9610042

If we want to perform ³⁄₂-copying for a qubit, we can, as a first step, embed the qubit $\alpha_0 |0\rangle + \alpha_1 |1\rangle$ isometrically into a qutrit and then produce three half-copies that are each qutrits. The decoding will result in a qutrit in state $\alpha_0 |0\rangle + \alpha_1 |1\rangle$ (where $\alpha_2 = 0$), which can, as a formality, be transformed back to a qubit by a projective measurement (specifically the projective measurement on the right side of figure 55).

12.2 More general forms of quantum redundancy

³⁄₂-copying is an example of a very simple type of quantum error-correcting code. The subject of quantum error-correcting codes, which can protect data against various types of errors, is explained further in sections 35 and 36 of *Part III: quantum information theory*.

Part II
Quantum algorithms

13 Classical and quantum algorithms as circuits

In this section, we'll see see a basic picture of classical and quantum algorithms as circuits. We'll consider simulations between classical and quantum circuits and we'll see the Toffoli gate.

13.1 Classical logic gates

Recall that the NOT gate takes one bit as input and outputs the logical negation of the bit.

a	$\neg a$
0	1
1	0

Figure 86: Table of input/output values of the NOT gate (symbolically denoted as $\neg$).

Here is the traditional notation for the gate that's used in electrical engineering logic diagrams, followed by more recent alternative ways of denoting the gate.

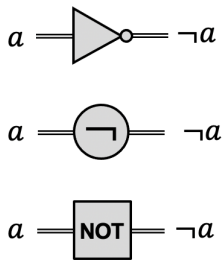

Figure 87: Three notations for the NOT gate.

The logical AND gate takes two input bits and produces one output bit, which is 1 if and only if both input bits are 1.

ab	$a \wedge b$
00	0
01	0
10	0
11	1

Figure 88: Table of input/output values of the AND gate (symbolically denoted as $\wedge$).

Here is the traditional electrical engineering notation for the AND gate followed by alternate notation.

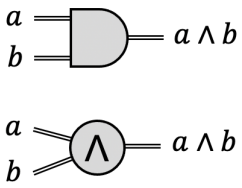

Figure 89: Two notations for the AND gate.

The *fan-out* operation is is often implicitly used in circuit diagrams. It's essentially a copying operation, whose output bits are multiple copies of its input bit. Recall that it is possible to copy classical information.

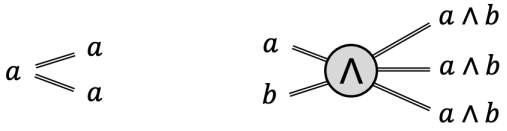

Figure 90: Notation for fan-out: of an input bit (left), and of the output of a gate (right).

What about the logical OR gate and the XOR gate? These are defined as

ab	$a \vee b$
00	0
01	1
10	1
11	1

ab	$a \oplus b$
00	0
01	1
10	1
11	0

Figure 91: Input/output values of the OR and XOR gates (denoted as $\vee$ and $\oplus$, respectively).

We don't need them as our fundamental operations because they can be *simulated* by $\wedge$ and $\vee$ gates. For example, OR can be simuated by three NOT gates and one AND gate using *DeMorgan's Law*

$$a \vee b = \neg(\neg a \wedge \neg b). \tag{144}$$

We can also simulate an XOR gate in terms of AND and NOT gates, which I leave as an exercise.

Exercise 13.1. Show that an XOR gate can be simulated by AND and NOT gates.

13.2 Computing the majority of a string of bits

Every binary string of odd length has either more zeroes than ones or more ones than zeroes. Define the *majority* of an odd-length string as the most common value. Here's a table of values of the majority function for 3-bit strings, denoted MAJ_3.

a	$\mathsf{MAJ}_3(a)$
000	0
001	0
010	0
011	1
100	0
101	1
110	1
111	1

Figure 92: Table of input/output values of the MAJ_3 function.

Here's a circuit that computes this majority function for 3-bit strings.

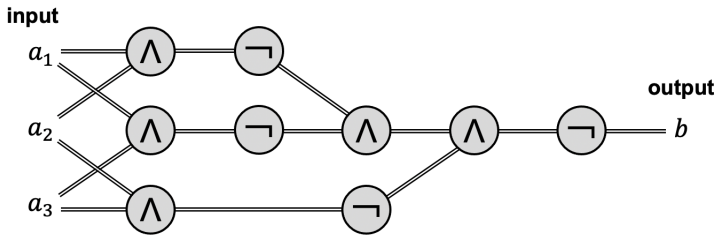

Figure 93: Classical circuit computing the majority value of three bits.

The input bits a_1, a_2, a_3 are on the left, information flows from left to right, and the output bit is $b = \mathsf{MAJ}_3(a_1, a_2, a_3)$. The circuit is based on this formula

$$\mathsf{MAJ}_3(a_1, a_2, a_3) = (a_1 \wedge a_2) \vee (a_1 \wedge a_3) \vee (a_2 \wedge a_3)\,, \tag{145}$$

where the OR gates are simulated by NOT and AND gates along the lines of Eq. (144). The total number of OR and NOT gates is nine (and there are three implicit fan-out gates at the inputs).

Can you construct a classical circuit for $\mathsf{MAJ}_n(a_1, a_2, \ldots, a_n)$, the majority value of $(a_1, a_2, \ldots, a_n)$, for odd $n > 3$? What is the gate count of your construction as a function of n? Does it grow polynomially or exponentially with respect to n?

Exercise 13.2 (level of difficulty depends on your prior familiarity with logic circuits)**.** Construct a classical circuit that computes $\mathsf{MAJ}_n(a_1, a_2, \ldots, a_n)$ for arbitrarily large odd n using a number of gates that scales polynomially with respect to n.

13.3 Classical algorithms as logic circuits

We can represent algorithms as logic circuits along the lines of this diagram.

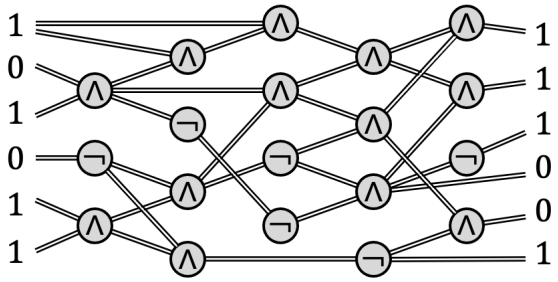

Figure 94: Generic form of a classical circuit (information flows from left to right).

The inputs are on the left, time flows left to right and the outputs are on the right. Classical computer algorithms are usually described in high level languages, but they implicitly represent many low-level logic operations, like this circuit.

A reasonable overall cost measure is the number of gates. There are other measures that we may care about such as the *depth*, which is the length of the longest path from an input to an output. Or the *width*, which is the maximum number of gates at any level, assuming that the gates are arranged in levels. But, to keep things simple, let's use the gate count as our main cost measure.

The number of gates depends on what the elementary operations are. We'll take these to be AND and NOT gates (and let's not count the fan-out operations). What's important is that *a gate does a constant amount of computational work*. If we allowed arbitrary gates then any circuit could be reduced to just one "supergate," but the cost of that one gate would not be very meaningful, since we expect large gates to be expensive to implement.

13.4 Multiplication problem and factoring problem

Let's consider the *multiplication problem* where one is given two n-bit binary integers as input and the output is their product (a $2n$-bit integer). For example, for inputs

101 (5 in binary) and 111 (7 in binary), the output should be 100011 (35 in binary) because the product of 5 and 7 is 35.

Think of the number of bits n as being quite large, larger than the number of bits of the arithmetic logic unit of your computer. Do you know of an algorithm for performing this? I believe that you do know such an algorithm, because you learned one in grade school. In principle, you could multiply two numbers consisting of thousands of digits by pencil and paper using that method. And it can be coded up as an algorithm (commonly referred to as the *grade school multiplication algorithm*). How efficient is this algorithm? Assuming you learned the algorithm that I did, its running time scales quadratically in its input size. By quadratic, I mean some constant times n^2 (for sufficiently large n).

The constant depends on the exact gate set that you use and we'll often use the *big-oh* notation to suppress that.

Definition 13.1 (big-oh notation)**.** For $f, g : \mathbb{N} \to \mathbb{N}$, $f(n) \in O(g(n))$ if $f(n) \leq cg(n)$ for some constant $c > 0$ (and for sufficiently large n).

Of course if we want to *implement* an algorithm then we do care about what the constant multiple is. But if we're looking at things theoretically then the big-oh notation helps reduce clutter and focus attention on *asymptotic* growth rates, which tend to be more important for large n.

There are more efficient algorithms than the grade school method. The current record is $O(n \log n)$ which is quite good. (There's a rich history of multiplication algorithms that evolved from $O(n^2)$ to $O(n^{1.585})$ to $O(n \log n \log \log n)$ to $O(n \log n)$, but we won't get into that here.)

Now, let's look at the *factoring problem*, which can be roughly thought of as the inverse of the multiplication problem. The input is an n-bit number and the output is its prime factors. For example, for input 100011, the outputs are 101 and 111 (the prime factors of 35).

You probably also learned this algorithm in grade school for factoring numbers: First check if the number is divisible by 2. Then check for divisibility by 3. You can skip 4, since that's a multiple of 2. Then check 5, skip 6, check 7, and so on. You can stop once you get to the square root of the number because if you haven't found a divisor by then, the number must be prime. To get a feel for this, try factoring the number 91 (or show that it's prime).

How expensive is this computationally? For large n, there are lots of divisors to check, exponentially many. The cost is exponential in n, namely $O(\exp(n))$. There are

more sophisticated algorithms than this method. The best currently-known classical algorithm for factoring is the so-called *number field sieve algorithm*, which scales exponentially in the cube root of n (to be more precise, $\exp(n^{1/3}\log^{2/3}(n))$). Since the cube root is in the exponent, this is still essentially exponential. There is no currently-known classical algorithm for factoring whose cost scales polynomially with respect to n.

In fact, the presumed difficulty of factoring is the basis of the security of many cryptosystems.

13.5 Quantum algorithms as quantum circuits

Now let's consider quantum circuits.

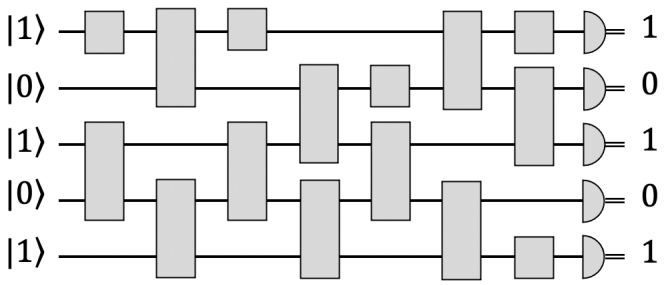

Figure 95: Generic form of a quantum circuit.

The input data is on the left, as computational basis states. Data flows from left to right as a series of unitary gates. Then measurement gates occur at the end, which yields the output of the computation.

Again, we can takes various cost measures for quantum circuits, such as the number of gates, the depth, or the width. Let's focus on the number of gates. As with classical circuits, we need a notion of what an *elementary* gate is. For now, let's consider the elementary gates to be the set of all 1-qubit and 2-qubit gates. We can consider simpler gate sets later on.

Consider the factoring problem again. Peter Shor's famous factoring algorithm can be expressed as a quantum circuit for factoring that consists of $O(n^2 \log n)$ gates—a polynomially bounded number of gates! An efficient implementation of this algorithm would break several cryptosystems, whose security is based in the presumed computational hardness of factoring.

14 Simulations between quantum and classical

It is natural to ask how the classical model of computation compares with the quantum model of computation. Can quantum circuits simulate classical circuits? Can classical circuits simulate quantum circuits? How efficient are the simulations?

For quantum circuits to simulate classical circuits, the Pauli X gate (that maps $|0\rangle$ to $|1\rangle$ and $|1\rangle$ to $|0\rangle$ can be viewed as a NOT gate, but what quantum gate corresponds to the AND gate? None of the operations that we've considered so far has any resemblance to the AND gate. The Toffoli gate makes such a connection.

14.1 The Toffoli gate

I'd like to show you a 3-qubit gate called the *Toffoli gate*, which will be useful for simulating AND gates. It's denoted this way, like a CNOT gate, but with an additional control qubit.

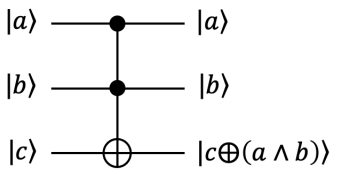

Figure 96: Toffoli gate acting on computational basis states ($a, b, c \in \{0, 1\}$).

On computational basis states, it flips the third qubit if and only if *both* of the first two qubits are in state $|1\rangle$; otherwise it does nothing. The 8×8 unitary matrix for this gate is

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}. \tag{146}$$

For arbitrary 3-qubit states, that are not necessarily computational basis states, the action of the gate on the state is to multiply the state vector by this 8×8 matrix.

The Toffoli gate is sometimes called the *controlled-controlled-NOT* gate. This makes sense, because the Toffoli gate is a controlled-CNOT gate.

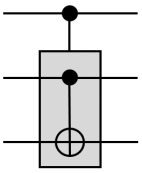

Figure 97: Toffoli gate is a controlled-CNOT gate (also a controlled-controlled-NOT gate).

14.2 Quantum circuits simulating classical circuits

Theorem 14.1. *Any classical circuit of size s can be simulated by a quantum circuit of size $O(s)$, where the gates used are Pauli X, CNOT, and Toffoli.*

Proof. We use Toffoli gates to simulate AND gates as illustrated in figure 98.

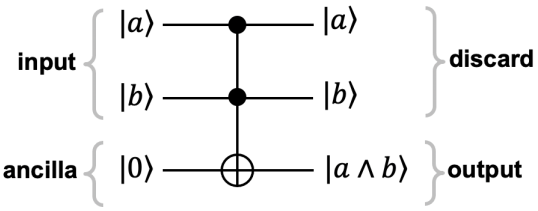

Figure 98: Using a Toffoli gate to simulate a classical AND gate.

Bits are represented as computational basis states in the natural way (bits $a, b \in \{0, 1\}$ are represented by $|a\rangle$ and $|b\rangle$). To compute the AND of a and b, a third ancilla qubit in state $|0\rangle$ is added and then a Toffoli gate is applied as shown in figure 98. This causes the state of the ancilla qubit to become $|a \wedge b\rangle$. The first two qubits can be discarded, or just ignored for the rest of the computation. That's how an AND gate can be simulated.

To simulate NOT gates, we can use the Pauli X gate.

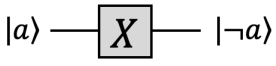

Figure 99: Using an X-gate to simulate a classical NOT gate.

Finally, we can explicitly simulate a fan-out gate by adding an ancilla in state $|0\rangle$ and apply a CNOT gate, as in figure 100.

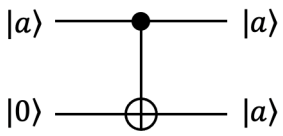

Figure 100: Using a CNOT gate to simulate a classical fan-out gate.

■

Remember that circuit in figure 93 for computing the majority of 3 bits? Here's a quantum circuit that simulates that circuit using Toffoli gates for AND and X gates for NOT.

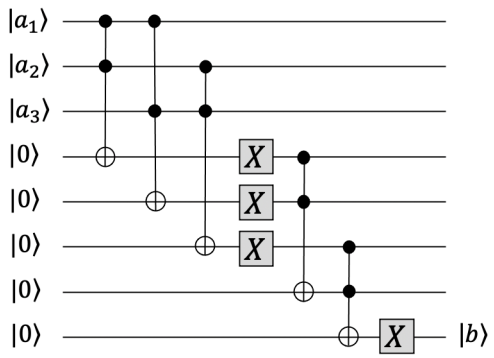

Figure 101: Computing the majority of three bits using Toffoli and X gates.

The output is the last qubit. (Note how this quantum circuit does not need to explicitly simulate the fan-out gates of the inputs.)

You may have noticed that we defined quantum circuits to be in terms of 1-qubit and 2-qubit gates, but our simulation of classical circuits used 3-qubit gates: the Toffoli gate. We can remedy this by simulating each Toffoli gate by a series 2-qubit gates. I'm going to show how to do this.

To start, we will make use of a 2×2 unitary matrix with the property that if you square it you get the Pauli X. Since X is essentially the NOT gate, this matrix is sometimes referred to as a "square root of NOT." Note that, in classical information, there is no square root of NOT, so the quantum square root of NOT can be regarded as a curiosity. Let's refer to this matrix as V.

Exercise 14.1 (straightforward)**.** Find a 2×2 unitary matrix V such that $V^2 = X$.

Henceforth, I'm going to assume that we have such a matrix V in hand. From this, we can define a controlled-V gate. Also, we can define a controlled-V^* gate. Now, consider the following circuit, consisting of 2-qubit gates.

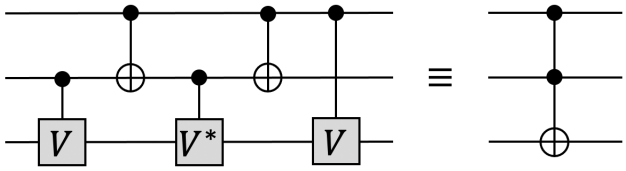

Figure 102: Simulating a Toffoli gate in terms of 2-qubit gates.

I claim that it computes the Toffoli gate. How can we verify this?

Let's discuss two possible approaches. The first approach has the advantage that it's completely mechanical. No creative idea is needed to carry it out. What you can do is work out the 8×8 matrix corresponding to each gate and then multiply[16] the five matrices and see if the product is the matrix in Eq. (146) for the Toffoli gate.

A second approach is to try to find shortcuts based on the logic of the gates, to avoid tedious calculations. In this case, note that it it sufficient to verify that the circuits are equivalent in the case where the first two qubits are in computational basis states $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$.

Consider the $|00\rangle$ case. In this case, none of the controlled-gates have any effect, so the state on the third qubit doesn't change—which is consistent with what the Toffoli gate does in this case. What about the $|01\rangle$ case? Then none of the gates controlled by the first qubit have any effect, so the circuit reduces to a controlled-V followed by a controlled-V^*, acting on the second and third qubits, which does not change the state (since $VV^* = I$). I'll leave it to you to analyze the other two cases.

Exercise 14.2. Continue the analysis of the correctness of the circuit in figure 102 for the cases where the control qubits are in state $|10\rangle$ and in state $|11\rangle$.

This kind of approach, when it can be made to work, has two advantages. First, it avoids a tedious calculation. Second, thinking about it this way, we can gain some intuition about why the circuit works. In the future, if you need to design a quantum circuit to achieve something, such intuition can be helpful.

Exercise 14.3 (Challenging)**.** Show how to simulate a Toffoli gate using only CNOT gates and 1-qubit gates. Thus, no controlled-U gates for $U \neq X$ are allowed.

Some classical algorithms make use of random number generators and we have not captured this in our definition of classical circuits. Say we allow our classical

[16] A word of caution: gates go from left to right, but the corresponding matrix products go from right to left (because we multiply vectors on the left by matrices).

algorithms to access random bits, that are zero with probability $\frac{1}{2}$ and one with probability $\frac{1}{2}$. Can we simulate such random bits with quantum circuits? This is actually quite easy, since constructing a $|+\rangle$ state and measuring it yields a random bit.

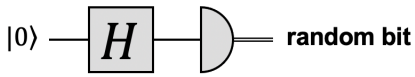

Figure 103: Using a Hadamard and a measurement gate to generate a classical random bit.

But this entails an intermediate measurement. Our quantum circuits will not look like the ones we defined earlier (of the form of figure 95), where all the measurements are at the end. Can we simulate random bits without the intermediate measurements? The answer is yes, by the following construction.

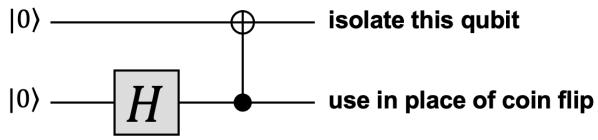

Figure 104: Simulating random bits without using measurements.

Instead of measuring the qubit in the $|+\rangle$ state, we apply a CNOT gate to a target qubit (an ancilla) in state $|0\rangle$. Then we ignore that ancilla qubit for the rest of the computation. The final probabilities when the circuit is measured at the end will be exactly the same as if a random bit had been inserted at that place in the computation. I will leave this as an exercise for you to prove that this works.

Using our decomposition of the Toffoli gate into 2-qubit gates and our results about simulating classical randomness, we can strengthen Theorem 14.1 to the following.

Corollary 14.1. *Any classical probabilistic circuit of size s can be simulated by a quantum circuit of size $O(s)$, consisting of 1-qubit and 2-qubit gates.*

14.3 Classical circuits simulating quantum circuits

Classical circuits can simulate quantum circuits but the only known constructions have exponential overhead.

Theorem 14.2. *A quantum circuit of size s acting on n qubits can be simulated by a classical circuit of size $O(sn^2 2^n)$.*

Proof. The idea of the simulation is quite simple. An n-qubit state is a 2^n-dimensional vector $\sum_{x\in\{0,1\}^n} \alpha_x |x\rangle$. Store the 2^n in an array of size 2^n, each with n-bits precision (which means accuracy within 2^{-n}).

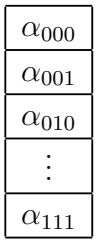

Figure 105: 2^n-dimensional array storing the amplitudes of state $\sum_{x\in\{0,1\}^n} \alpha_x |x\rangle$.

The initial state is a computational basis state. Each gate corresponds to a $2^n \times 2^n$ matrix and the effect of the gate is to multiply the state vector by that matrix. In fact that matrix will be sparse for 1-qubit gates and 2-qubit gates, with only a constant number of nonzero entries for each row and each column. Therefore, there are a constant number of arithmetic operations per entry of the array. Let's allow $O(n^2)$ elementary gates for each such arithmetic operation[17] (on n-bit precision numbers). The simulation cost is $O(n^2 2^n)$ for each gate in the circuit. We multiply that by the number of gates in the quantum circuit s to get a cost of $O(sn^2 2^n)$. That's the gate cost to get the final state of the computation, just before the measurement.

What about the measurements? For this, we compute the absolute value squared of each entry of the array. Again, that's 2^n arithmetic operations. At this point, the entries in the array are the probabilities after the measurement.

The output of the quantum circuit is not the probabilities; it's a sample according to the distribution. How do we generate that sample? First we sample the first bit of the output, the probability that that this bit is 0 is the sum of the first half of the entries of the array; the probability that bit is 1 is the sum of the second half. Once we have the first bit, we can use a similar approach for the second bit, using conditional probabilities, conditioned on the value of the first bit, and so on for the other bits. This also costs $O(n^2 2^n)$ elementary classical gates. ■

Note that if we take the quantum circuit that results from Shor's algorithm and then simulate it by a classical circuit then simulate that quantum circuit by a classical

[17] Using simple grade-school algorithms for addition and multiplication.

circuit, the result is not very efficient. It's exponential and in fact it's worse than the best currently-known classical algorithm for factoring.

If we view the aforementioned simulation in the usual high-level language of algorithms, it is exponential *space* in addition to exponential time (because it stores the huge array). In fact there's a way to reduce the storage space to polynomial—while maintaining exponential time.

Exercise 14.4 (challenge)**.** Show how to do the above simulation as an algorithm using only a polynomial amount of space (memory).

15 Computational complexity classes in brief

In theoretical computer science, there are various taxonomies of computational problems according to their computational difficulty. I'll briefly show you a few of these and how the power of quantum computers fits in within them.

Consider all binary functions over the set of all binary strings (of the form $f : \{0,1\}^* \to \{0,1\}$, where $*$ denotes the Kleene closure[18] in this context). The input is a binary string of some length n, where n can be anything. And the output is one bit. These are sometimes called decision problems since the answer is a binary decision, 0 or 1, yes or no, accept or reject. This is a common convention for reasons of simplicity. Most problems that are not naturally decision problems can be reworked to be expressed as decision problems.

P (polynomial time)
Solvable by $O(n^c)$-size classical circuits[19] (for some constant c).

BPP (bounded-error probabilistic polynomial time)
Solvable by $O(n^c)$-size probabilistic classical circuits whose worst-case error probability[20] is $\leq \frac{1}{4}$.

BQP (bounded-error quantum polynomial time)
Solvable by $O(n^c)$-size quantum circuits with worst-case error probability $\leq \frac{1}{4}$.

EXP (exponential time)
Solvable by $O(2^{n^c})$-size classical circuits.

The following containments among these complexity classes are known:

$$\mathbf{P} \subseteq \mathbf{BPP} \subseteq \mathbf{BQP} \subseteq \mathbf{EXP}. \tag{147}$$

[18] More specifically, $\{0,1\}^* = \varnothing \cup \{0,1\} \cup \{0,1\}^2 \cup \{0,1\}^3 \cup \cdots$.

[19] Technically, we require *uniform circuit families*, one for each input size n, with an algorithmic way of mapping n to the circuit for size n.

[20] There is some arbitrariness in the error bound $\frac{1}{4}$. Any polynomial-size circuit achieving this can be converted into another circuit whose error probability is $\leq \epsilon$ by repeating the process $\log(1/\epsilon)$ times and taking the majority value. Using $\frac{1}{4}$ is simple, though any constant below $\frac{1}{2}$ would work.

16 The black-box model

In the next few subsequent sections, we are going to see some simple quantum algorithms in a framework called the *black-box model*. First, in this section, I'll explain this computational model.

16.1 Classical black-box queries

Imagine that f is some function that is unknown to us and we're given a device that enables us to evaluate f on any particular input, of our choosing, and that this is our *only* way of acquiring information about f. Such a device is commonly called a *black-box* for f.

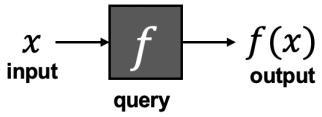

Figure 106: A gate performing an f-query.

If we want to evaluate the function on some input x, we insert x into the black-box and then the value of $f(x)$ pops out, for us to see. We call this process of evaluating the function at an input an *f-query*.

Now, suppose that we want to acquire some specific information about a function f, and that we want to do this with as few queries as possible. We can think of this as a game, where we want to know something about an unknown function f, and we're only allowed to ask questions like "what's the value of f at point x?" for any x of our choosing. How many such questions do we have to ask?

One example that illustrates the general idea is *polynomial interpolation*. Here, one is given a black-box computing an unknown polynomial of degree up to d, and the goal is to determine which polynomial it is. At how many points does the polynomial have to be evaluated to accomplish this?

Figure 107: Interpolating linear and quadratic functions with as few queries as possible.

If $f : \mathbb{R} \to \mathbb{R}$ is an unknown linear function then one evaluation is insufficient for determining what f is; however, two queries suffice (linear interpolation). If $f : \mathbb{R} \to \mathbb{R}$ is quadratic then it turns out that three evaluations are necessary and sufficient. In general, if f is a polynomial with degree up to d then the number of queries that are necessary and sufficient is $d + 1$.

We will focus our attention on functions over *finite* domains instead of $\mathbb{R}$, such as $\{0,1\}^n$. Let us begin by considering the simple case where we have an unknown $f : \{0,1\} \to \{0,1\}$. There are only four such functions and here are the tables of values for each of them:

x	$f(x)$
0	0
1	0

x	$f(x)$
0	1
1	1

x	$f(x)$
0	0
1	1

x	$f(x)$
0	1
1	0

Figure 108: The four functions $f : \{0,1\} \to \{0,1\}$.

Suppose that we're given a black-box for such a function, but we don't know of the four functions it is.

Suppose that our goal is to determine whether:

- $f(0) = f(1)$ (the first two cases in figure 108); or
- $f(0) \neq f(1)$ (the last two cases in figure 108).

To be clear, we are not required to determine which of the four functions f is; just whether it's among the first two or the last two. How many queries do we need to accomplish this? Please think about this. We will get back to this question in section 17.1.

16.2 Quantum black-box queries

A classical query is along the lines of figure 106. We can set the input to any x in the domain of f. Then we receive as output the value of $f(x)$. Does it make sense to define a *quantum* query? Let's keep our attention on the simple case where $f : \{0,1\} \to \{0,1\}$ (figure 108); we will consider more general cases later.

I first want to show you a naïve first attempt at defining a quantum query that does *not* work. The classical query maps bits to bits. Define a quantum query (mapping qubits to qubits) that correspondingly maps computational basis states to computational basis states, as in figure 109.

Figure 109: Naïve attempt to define a quantum query—that doesn't work!

For each of the four functions $f : \{0,1\} \to \{0,1\}$ in figure 108, here are the corresponding mappings on computational basis states.

$\|a\rangle$	$\|f(a)\rangle$
$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$\|0\rangle$

$\|a\rangle$	$\|f(a)\rangle$
$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$

$\|a\rangle$	$\|f(a)\rangle$
$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$\|1\rangle$

$\|a\rangle$	$\|f(a)\rangle$
$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$

Figure 110: Input-output relationships for naïvely defined quantum query gate.

By linearity, we get these four linear operators.

$$\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{148}$$

The third and fourth are familiar unitary operations: the identity operation and the Pauli X operation. But notice that the first two are not unitary operations. This violation of unitarity is a serious problem. For example, the first two linear operators both map the state $|-\rangle$ to the zero vector, a two-dimensional vector whose amplitudes are both zero, which makes no sense as a quantum state. So this approach does not work.

In order for a classical mapping to be quantizable in the above manner it must be bijective. Then the underlying linear operator is given by a permutation matrix, which is unitary.

We will first define a *reversible classical f-query* that is bijective whether or not f itself is bijective. In the case where $f : \{0,1\} \to \{0,1\}$, the reversible classical f-query is the mapping $\{0,1\}^2 \to \{0,1\}^2$ defined as $(a,b) \mapsto (a, b \oplus f(a))$ (for all $a, b \in \{0,1\}$). Here is notation for a reversible classical f-query.

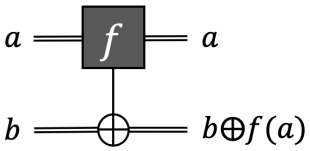

Figure 111: Reversible classical f-query (for $f : \{0,1\} \to \{0,1\}$).

Why is this mapping $\{0,1\}^2 \to \{0,1\}^2$ bijective? One way to see this is to observe that the mapping is its own inverse.

The reversible classical f-query is easy to quantize as the 2-qubit unitary operation that maps $|a\rangle|b\rangle$ to $|a\rangle|b \oplus f(a)\rangle$ (for all $a, b \in \{0,1\}^2$). Here is notation for a quantum f-query (in the case where $f : \{0,1\} \to \{0,1\}$).

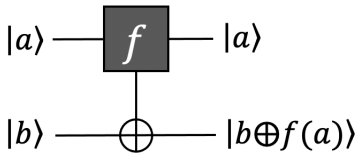

Figure 112: Quantum f-query (for $f : \{0,1\} \to \{0,1\}$).

Note that the above defines the effect of the f-query on computational basis states. This determines a unitary operator that defines the effect of the f-query on arbitrary quantum states.

We can generalize the above definition to arbitrary functions $f : \{0,1\}^n \to \{0,1\}^m$. The f-query is a unitary operation acting on $n+m$ qubits defined as follows.

Definition 16.1. Let function $f : \{0,1\}^n \to \{0,1\}^m$. Then a *quantum f-query* is defined as the unitary operation acting on $n+m$ qubits with the property that, for all $a \in \{0,1\}^n$ and $b \in \{0,1\}^m$,

$$|a\rangle|b\rangle \mapsto |a\rangle|b \oplus f(a)\rangle, \tag{149}$$

where $b \oplus f(a)$ denotes the bit-wise[21] XOR between the m-bit strings b and $f(a)$.

Here is notation for a general quantum f-query.

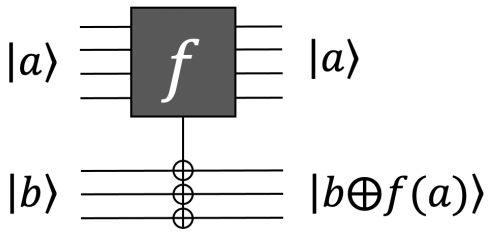

Figure 113: Quantum f-query (for $f : \{0,1\}^n \to \{0,1\}^m$).

[21] The *bit-wise* XOR between $(b_1, b_2, \ldots, b_m)$ and $(c_1, c_2, \ldots, c_m)$ is $(b_1 \oplus c_1, b_2 \oplus c_2, \ldots, b_m \oplus c_m)$.

17 Simple quantum algorithms in black-box model

The simple quantum algorithms in this section are admittedly curiosities, rather than practical algorithms. It's hard to think of real-world applications that fit their framework, and where it's worth the trouble to build a quantum device to solve these problems. What you should pay attention to are the maneuvers that these quantum algorithms make. After these algorithms, we'll be seeing increasingly sophisticated extensions of these maneuvers, that accomplish more dramatic algorithmic feats.

17.1 Deutsch's problem

The first problem that we'll consider involves functions $f : \{0,1\} \to \{0,1\}$ (the four such functions are shown in figure 108). Remember the question of how many queries are necessary to determine whether or not $f(0) = f(1)$? This is the definition of Deutsch's Problem.[22]

Definition 17.1. *Deutsch's Problem* is defined as the problem where one is given as input a black box for some $f : \{0,1\} \to \{0,1\}$ and the goal is to determine whether or not $f(0) = f(1)$ by making queries to f.

Let's first consider classical queries necessary to solve Deutsch's problem. One query is not sufficient. To see why this is so, suppose that you make one query at some $a \in \{0,1\}$ to acquire $f(a)$. This gives absolutely no information about the *other* value, $f(\neg a)$. It is possible that $f(\neg a) = f(a)$ and it is possible that $f(\neg a) \neq f(a)$. Therefore, two queries necessary and clearly two queries are also sufficient.

You may wonder whether the classical query cost is different when *reversible* classical queries are used. In fact, reversible classical query at (a, b) provide exactly the same amount of information as a simple classical queries of at a. The output of the reversible query at (a, b) is $(a, b \oplus f(a))$, and note that (a, b) are already known. Therefore, there are only two possibilities of interest for the output:

$$\begin{cases} b \oplus f(a) = b, \text{ which occurs if any only if } f(a) = 0; \\ b \oplus f(a) \neq b, \text{ which occurs if any only if } f(a) = 1. \end{cases} \tag{150}$$

Therefore, even with reversible classical queries, one query is not sufficient.

[22] D. Deutsch (1985). "Quantum theory, the Church-Turing principle, and the universal quantum computer." *Proc. R. Soc. London A*, **400**(1818): 97–117.

Now let's see what an algorithm solving Deutsch's Problem with two reversible classical queries looks like. Here's a classical circuit expressing such an algorithm.

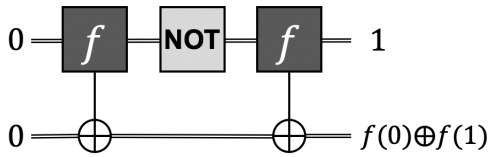

Figure 114: Classical algorithm for Deutsch that makes two reversible classical queries.

The first query XORs the value of $f(0)$ to the second bit. Then the first bit is flipped to 1, so the second query XORs the value of $f(1)$ to the second bit. At the end, the second bit contains the value of $f(0) \oplus f(1)$, which is the solution to the problem.

There is an obvious 2-query quantum algorithm that solves Deutsch's problem just like the circuit in figure 114. But quantum circuits need not be restricted to these types of operations, where states are always in computational basis states. This quantum circuit that solves Deutsch's problem with just one single query!

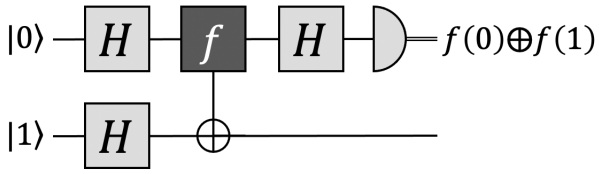

Figure 115: Quantum algorithm for Deutsch that makes just one quantum query.

How does it work? It's very different from any classical algorithm and we'll analyze it carefully. There are three Hadamard transforms, and each one plays a different role in the computation. Let's look at how each Hadamard contributes to the computation, in the order shown in figure 116.

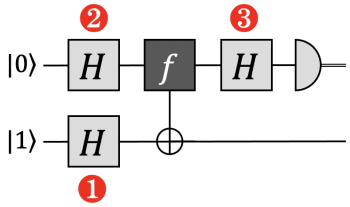

Figure 116: The three separate Hadamard transforms.

❶ What the first Hadamard does

This is the Hadamard applied to the second qubit, in state $|1\rangle$. Obviously, this creates the $|-\rangle$ state. What's interesting about creating this state here is that it's an eigenvector of the Pauli X. Performing an X-operation on $|-\rangle$ causes it to become[23] $\frac{1}{\sqrt{2}}|1\rangle - \frac{1}{\sqrt{2}}|0\rangle = -|-\rangle$.

With the second qubit in state $|-\rangle$, if the first qubit is in the computational basis state $|a\rangle$ then the f-query causes the second qubit to change by a factor of -1 if and only if $f(a) = 1$, as shown in the following circuit diagram.

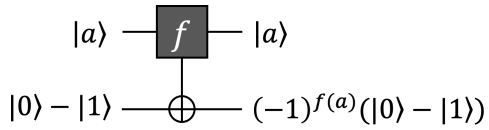

Figure 117: Caption.

Note that $(-1)^{f(a)}$ is a succinct notation for expressing the two cases, since

$$(-1)^{f(a)} = \begin{cases} +1 & \text{if } f(a) = 0 \\ -1 & \text{if } f(a) = 1. \end{cases} \tag{151}$$

Notice that the 2-qubit output state is $(-1)^{f(a)}|a\rangle|-\rangle$ and the $(-1)^{f(a)}$ does not really belong to a specific qubit of the two. We can equivalently write the circuit this way.

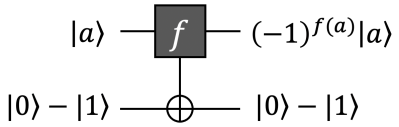

Figure 118: Caption.

But this is an interesting way of thinking about what the query does when the second qubit is in state $|-\rangle$. Namely, the first qubit picks up a phase of $(-1)^{f(a)}$, and the second qubit doesn't change. We sometimes call this *querying in the phase*.

At this point, you might wonder why we should care about this, since this is a global phase, and we know that global phases don't matter. But it's only a global phase if the first qubit is in a computational basis state, of the form $|a\rangle$. It's *not* a global phase if the first qubit is in superposition. This brings us to the role of the second Hadamard.

[23] Let's keep track of the distinction between between $|-\rangle$ and $-|-\rangle$, even though it's only a global phase. The significance of this will become clear shortly.

❷ What the second Hadamard does

This brings us to the role of the second Hadamard, that's applied to the first qubit. That causes the first input qubit to the query to be in an equally weighted superposition of $|0\rangle$ and $|1\rangle$, namely, the $|+\rangle$ state. Now the state of the first output qubit after the query is in a superposition of $|0\rangle$ and $|1\rangle$ with respective phases $(-1)^{f(0)}$ and $(-1)^{f(1)}$.

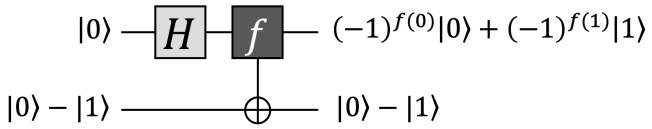

Figure 119: Caption.

So, after the query, the state of the first qubit is $\frac{1}{\sqrt{2}}(-1)^{f(0)}\,|0\rangle + \frac{1}{\sqrt{2}}(-1)^{f(1)}\,|1\rangle$. Let's look at what this state is for each of the four possible functions back in figure 108. The corresponding states of the first qubit are respectively

$$\tfrac{1}{\sqrt{2}}\,|0\rangle + \tfrac{1}{\sqrt{2}}\,|1\rangle \qquad -\tfrac{1}{\sqrt{2}}\,|0\rangle - \tfrac{1}{\sqrt{2}}\,|1\rangle \qquad \tfrac{1}{\sqrt{2}}\,|0\rangle - \tfrac{1}{\sqrt{2}}\,|1\rangle \qquad -\tfrac{1}{\sqrt{2}}\,|0\rangle + \tfrac{1}{\sqrt{2}}\,|1\rangle\,. \tag{152}$$

Notice that, for the first two cases (where $f(0) = f(1)$), the state is $\pm\Big(\frac{1}{\sqrt{2}}\,|0\rangle + \frac{1}{\sqrt{2}}\,|1\rangle\Big)$; and for the last two cases (where $f(0) \neq f(1)$), the state is $\pm\Big(\frac{1}{\sqrt{2}}\,|0\rangle - \frac{1}{\sqrt{2}}\,|1\rangle\Big)$. Therefore, to solve Deutsch's problem, we need only distinguish between $\pm\,|+\rangle$ and $\pm\,|-\rangle$. Which brings us to the third Hadamard.

❸ What the third Hadamard does

This Hadamard maps $\pm\,|+\rangle$ to $\pm\,|0\rangle$ and $\pm\,|-\rangle$ to $\pm\,|1\rangle$. Measuring the resulting qubit in the computational basis yields

$$\begin{cases} 0 & \text{if } f(0) = f(1) \\ 1 & \text{if } f(0) \neq f(1). \end{cases} \tag{153}$$

Therefore, the output bit of the algorithm is the solution to Deutsch's problem.

Figure 120 summarizes the role that each of the three Hadamard transforms plays in the 1-query quantum algorithm for Deutsch's problem.

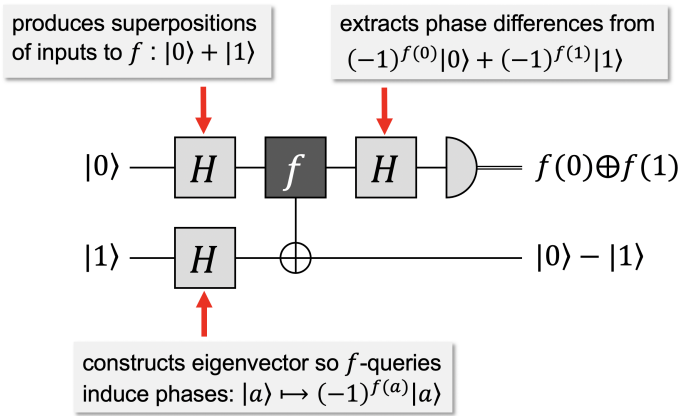

Figure 120: Summary of the role that each H transformation plays in the algorithm.

This quantum algorithm accomplishes something with one query that would require two classical queries by any classical algorithm.

Notice what this algorithm does *not* do. It does not somehow extract both values of the function, $f(0)$ and $f(1)$, with one single query. In fact, if you run this algorithm, then you get *no* information about the value of $f(0)$ itself. Also, you get *no* information about the value $f(1)$ itself. You *only* get information about $f(0) \oplus f(1)$.

17.2 One-out-of-four search

Now let's try to generalize this methodology to another problem.

Let $f : \{0,1\}^2 \to \{0,1\}$ with the special property that it attains the value 1 at exactly one of the four points in its domain. There are four possibilities for such a function and here are their truth tables.

x	$f_{00}(x)$
00	1
01	0
10	0
11	0

x	$f_{01}(x)$
00	0
01	1
10	0
11	0

x	$f_{10}(x)$
00	0
01	0
10	1
11	0

x	$f_{11}(x)$
00	0
01	0
10	0
11	1

Figure 121: The four functions $f : \{0,1\}^2 \to \{0,1\}$ that take value 1 at a unique point.

Now, suppose that you're given a black-box computing such a function. You're promised that it's one of these four, but you're not told which one. Your goal is to determine which of the four functions it is.

First of all, how many classical queries do you need to do this? The answer is three queries. You can query in the first three places and see if one of them evaluates to 1. If none of them do then, by process of elimination, you can deduce that the 1 must be the fourth place.

Now, what about quantum queries? Let's try to build a good quantum algorithm for this. For functions mapping two bits to one bit, the queries look like this.

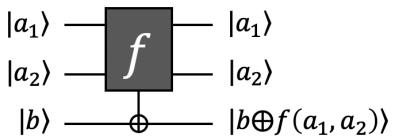

Figure 122: Query gate for a function $f : \{0,1\}^2 \to \{0,1\}$.

There are two qubits for the input, and a third qubit for the output of the function. In the computational basis, the value of $f(a_1, a_2)$ is XORed onto the third qubit. Of course, that description is for states in the computational basis. But, there is a unique unitary operation that matches this behavior on the computational basis states.

Let's start, along the lines that we did for Deutsch's algorithm: by setting the target qubit to state $|-\rangle$ so as to query in the phase; and then querying the inputs in a uniform superposition.

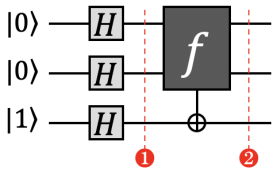

Figure 123: Starting along the lines of Deutsch's algorithm. (Intermediate stages are marked.)

Let's trace through the execution of this quantum circuit. At each stage, we will determine the state of the three qubits at that stage.

State at stage ❶

The state after the Hamadard operations, but just before the query is

$$\Big(|00\rangle + |01\rangle + |10\rangle + |11\rangle\Big)\,|-\rangle\,. \tag{154}$$

State at stage ❷

The query does not affect the state of the third qubit, but it changes the state of the first two qubits to

$$\begin{cases} -|00\rangle + |01\rangle + |10\rangle + |11\rangle & \text{in the case of } f_{00} \\ |00\rangle - |01\rangle + |10\rangle + |11\rangle & \text{in the case of } f_{01} \\ |00\rangle + |01\rangle - |10\rangle + |11\rangle & \text{in the case of } f_{10} \\ |00\rangle + |01\rangle + |10\rangle - |11\rangle & \text{in the case of } f_{11}. \end{cases} \tag{155}$$

Looking at these four states, can you think of a noteworthy property that they have? Please pasue to think about this ...

The states are orthogonal! If you take the dot-product between any two of these vectors then you get two positive terms and two negative terms, which cancel out. Since they're orthogonal, we can measure with respect to this basis. In other words, there exists a unitary operation U that maps these states to the computational basis, and then we can measure in the computational basis.

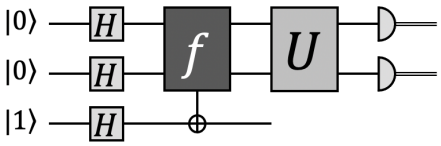

Figure 124: Quantum algorithm for the 1-out-of-4 search problem.

So this quantum circuit only makes one quantum query and it correctly identifies the function (among the four). And this would require 3 classical queries.

What is U? It's the 4×4 matrix whose rows are the vectors in Eq. (155).

An interesting question is whether it's possible to get a more dramatic quantum vs. classical query separation than 1 vs. 3.

17.3 Constant vs. balanced

Next we'll see a problem where a quantum algorithm solves a problem with exponentially fewer queries than any classical algorithm.[24]

[24] D. Deutsch and R. Jozsa (1992). "Rapid solutions of problems by quantum computation." *Proc. of the Royal Society of London A*, **439**(1907): 553–558.

Let $f : \{0,1\}^n \to \{0,1\}$. We call such a function *constant* if either its value is 0 everywhere, or its value is 1 everywhere. We call such a function *balanced* if its value is 0 in exactly the same number of places that its value is 1. There are 2^n possible inputs to such a function. Therefore, a balanced function takes value 0 in exactly 2^{n-1} places and it takes value 1 in exactly 2^{n-1} places.

Here are some examples of functions for the $n = 3$ case:

a	$f_1(a)$	$f_2(a)$	$f_3(a)$	$f_4(a)$	$f_5(a)$
000	0	1	0	1	0
001	0	1	0	1	0
010	0	1	0	0	0
011	0	1	0	1	0
100	0	1	1	0	0
101	0	1	1	0	0
110	0	1	1	0	1
111	0	1	1	1	0

Figure 125: Examples of functions that are constant, balanced, and neither.

The first two functions, f_1 and f_2, are the two constant functions: the all-zero function and the all-one function. The third function f_3 is a balanced function with all the zeroes before all the ones. And f_4 is another balanced function, but with the positions of the zeros and ones mixed up. How many balanced functions are there? Exponentially many (the number is approximately $\frac{2^n}{\sqrt{n}}$). And f_5 is neither constant nor balanced—just as a reminder that this third category exists.

For the *constant-vs-balanced problem*, we're given a black-box computing a function that is promised to be either constant or balanced, but we're not told which one. Our goal is to figure out which of the two cases it is, with as few queries to f as possible.

First of all, how many queries do we need to solve this problem by a classical algorithm? If you think about this, you'll probably realize that, even if you query the function in many spots and always see the same value, you still might not know which way it goes. It could be constant. But it could also be that, in many of the places that you did not query, the function takes the opposite value and it's actually a balanced function. It's only after you've queried in more than half of the spots that you can be sure about which case it is. Therefore, the number of queries that a classical algorithm must make is $2^{n-1} + 1$. That's a lot of queries when n is large.

How many queries can a quantum algorithm get by with? In fact, this problem can be solved by a quantum algorithm that makes just one single query! Let's see how that works.

We'll start off as usual, setting the target qubit to state $|-\rangle$ and setting the the other qubits—those where the inputs to f are—into a uniform superposition of all n-qubit basis states. That's what applying a Hadamard to each of n qubits in state $|0\rangle$ does.

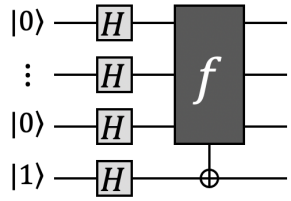

Figure 126: Starting off in a manner similar to the two previous algorithms.

So the state right after the query is

$$\frac{1}{\sqrt{2^n}}\left(\sum_{a\in\{0,1\}^n}(-1)^{f(a)}\,|a\rangle\right)\otimes|-\rangle\,. \tag{156}$$

The first n qubits are the interesting part of this state, which is a uniform superposition of all 2^n computational basis states of n qubits, with a phase of $(-1)^{f(a)}$ for each $|a\rangle$.

What does this state looks like in the constant case? In that case, either all the phases are $+1$ or all the phases are -1. So the state remains in a uniform superposition of computational basis states and just picks up a global phase of $+1$ or -1.

Now, what can we say about the state in the balanced case? There are lots of possibilities, depending on which particular balanced function it is. But one thing we can say is that the state is orthogonal to the state that arises in the constant case. Why? Because, in the computational basis, the state will have $+1$ in half of its components and -1 in the other half. So when you take the dot product, with a vector that has (say) $+1$ in each component you get a sum of an equal number of $+1$s and -1s, which cancel and results in zero.

Something that we've seen a few times before is that, when the cases that we're trying to distinguish between result in orthogonal states, we're in good shape. Being orthogonal means that, in principle, the states are perfectly distinguishable.

Let's make this more explicit. Suppose that we now apply a Hadamard to each of the first n qubits, and consider what happens in the constant case and in the balanced

case. In the constant case, this transforms the state to $\pm|00\ldots0\rangle = \pm|0^n\rangle$. In the balanced case, since unitary operations preserve orthogonality relationships, the state is transformed to some state that is orthogonal to $|0^n\rangle$.

After this final layer of Hadamards, we measure the state of the first n qubits.

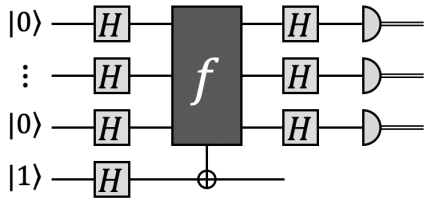

Figure 127: Quantum algorithm for the constant-vs-balanced problem.

If the outcome is $00\ldots0 = 0^n$ then we report "constant" and if the outcome is any string that's not 0^n (not all zeroes) then we report "balanced". Note that, in the balanced case, it's impossible to get outcome 0^n, because measuring a state orthogonal to $|0^n\rangle$ cannot result in outcome 0^n.

That's how the one-query quantum algorithm for the constant-vs-balanced problem works. It achieves something with one single query that requires exponentially many queries by any classical algorithm.

17.4 Probabilistic vs. quantum query complexity

Although everything I've said about the classical and the quantum query cost for the constant-vs-balanced problem is true, there's something unsatisfying about the "exponential reduction" in the number of queries that the quantum algorithm attains. The classical query cost is expensive *only if we require absolutely perfect performance.* If a classical procedure queries f in random locations then, in the case of a balanced function, it would have to be very unlucky to always draw the same bit value.

Here's a classical probabilistic procedure that makes just two queries and performs fairly well. It selects two locations randomly (independently) and then outputs "constant" if the two bits are the same and "balanced" if the two bit values are different.

What happens if f is constant? In that case the algorithm always succeeds. The two bits will always be the same. What happens if f is balanced? In that case the algorithm succeeds with probability $\frac{1}{2}$. The probability that the two bits will be different will be $\frac{1}{2}$.

By repeating the above procedure k times, we can make the error probability exponentially small with respect to k. Only 4 queries are needed to obtain success probability $\frac{3}{4}$. And, the success probability can be a constant that's arbitrarily close to 1, using a constant number of queries.

In summary, we've considered three problems in the black-box model. For each problem, a quantum algorithm solves it with just one query, but more queries are required by classical algorithms.

problem	quantum	classical deterministic	classical probabilistic
Deutsch	1	2	2
1-out-of-4 search	1	3	3
Const. vs. balanced	1	$2^{n-1}+1$	$O(1)$

Figure 128: Summary of query costs for problems considered so far.

For Deutsch's problem, any classical algorithm requires 2 queries. For the 1-out-of-4 search problem, any classical algorithm requires 3 queries. And, for the constant-vs-balanced problem, any classical algorithm requires exponentially many queries to solve the problem perfectly; however, for an arbitrarily small constant $\varepsilon > 0$, there is a probabilistic classical algorithm that makes only a constant number of queries and solves the problem with error probability less than ε.

Along this line of thought, the following question seems natural: Is there a black-box problem for which the quantum-vs-classical query cost separation is stronger? For example, for which even probabilistic classical algorithms with small constant error probability require exponentially more queries than a quantum algorithm?

We'll address this question in the next section.

18 Simon's problem

We are going to investigate a black-box problem called *Simon's Problem.*[25]

For this problem, there is an exponential difference between the probabilistic classical query cost and the quantum cost. Also the quantum algorithm for this problem introduces some powerful algorithmic techniques in a simple form. Simon's Problem is a slightly more complicated black-box problem than those that we've seen up to now, involving functions from n bits to n bits.

It is interesting for two reasons:

1. It improves on the progression of black-box problems where quantum algorithms outperform classical algorithms. The quantum algorithm requires exponentially fewer queries than even probabilistic classical algorithms that can err with constant probability, say $\frac{1}{4}$.
2. The quantum algorithm for Simon's problem introduces a technique that transcends the black-box model. When looked at the right way, the ideas introduced in Simon's algorithm lead naturally to Shor's algorithm for the discrete log problem—which is not a black-box problem! Shor discovered his algorithms (for discrete log and factoring) soon after seeing Simon's algorithm, and in his paper, he acknowledges that he was inspired by Simon's algorithm.

18.1 Definition of Simon's Problem

The problem concerns functions $f : \{0,1\}^n \to \{0,1\}^n$ that are *2-to-1 functions*, which means that, for every point in the range, there are exactly two pre-images. In other words, for every value that the function attains, there are exactly two points, a and b, in the domain that both map to that value. We call such a pair of points a *colliding pair*. If $a \neq b$ and $f(a) = f(b)$ then a and b are a colliding pair.

Now, there's a special property that some 2-to-1 functions have, that we'll call the *Simon property.*

Definition 18.1. A 2-to-1 function $f : \{0,1\}^n \to \{0,1\}^n$ has the *Simon property* if there exists $r \in \{0,1\}^n$ such that: $a \oplus b = r$, for every colliding pair (a, b).

[25] D. R. Simon (1994). "On the power of quantum computation." *Proc. 35th Annual Symp. on Foundations of Computer Science*, pp. 116–123.

For $a, b \in \{0,1\}^n$, $a \oplus b$ denotes their bit-wise XOR. That is, you take the XOR of the first bit of a with the first bit of b, and then you take the XOR of the second bit of a with the second bit of b, and so on.

Let's look at an example. Here's a 2-to-1 function $f : \{0,1\}^3 \to \{0,1\}^3$.

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

Figure 129: Example of function satisfying the Simon property for $n = 3$.

I've color coded the colliding pairs. For example, if you look at the two green points in the domain, 000 and 101, you can see that f maps both of these points to 011 (and those are the only points that are mapped to 011). So the two green points are a colliding pair. Also, the two red points 011 and 110 are a colliding pair, both mapping to 010. And there is a blue colliding pair and a purple colliding pair.

OK, so this f is a 2-to-1 function. But it also has the additional Simon property. If you take any colliding pair (a, b) then $a \oplus b$ is always the same 3-bit string. Can you see what that string r is in this example?

Did you get $r = 101$? If you pick any color and take the bit wise XOR of the two strings of that color you'll get 101. For example, for the red pair, $011 \oplus 110 = 101$.

Note that, for an arbitrary 2-to-1 function, the bit-wise XORs can be different for different colliding pairs. So the 2-to-1 functions that satisfy the Simon property are special ones, for which these bit-wise XORs are always the same.

Now we can define Simon's problem.

Definition 18.2 (Simon's problem)**.** You are given access to a black-box for a 2-to-1 function $f : \{0,1\}^n \to \{0,1\}^n$ that has the Simon property, but you are given no other information about f. You don't know what the colliding pairs are and you don't know the value of r. Your goal is to find the value of r.

In the above example, $r = 101$. For a general function $f : \{0,1\}^n \to \{0,1\}^n$, r could potentially be any non-zero n-bit string.

18.2 Classical query cost of Simon's problem

Let's try to think about how hard this black-box problem is. In the example, we could find r by taking the bit-wise XOR of any colliding pair. So, once we have a colliding pair, it's easy to find r. Therefore, one way to solve this is to find a colliding pair. But notice that any two distinct n-bit strings *could* be a colliding pair for some r. So if we make a query at some point a, we learn the value of $f(a)$, but we have no idea for which $b \neq a$, you get a colliding pair.

Here's an example of a classical algorithm for Simon's problem. If we a have a budget of m queries, query f at m random points in $\{0,1\}^n$ and then check if there's a collision among them. If any two are a colliding pair then their bit-wise XOR is the answer r. If there are no collisions then the algorithm is out of luck; in that case it fails to find r.

How large should m be set to so that the probability of a collision is at least $\frac{3}{4}$? There are 2^n points in the domain, and each pair of queries will collide with probability $\frac{1}{2^n}$. Since there are around m^2 pairs, the expected number of collisions is roughly m^2 times $\frac{1}{2^n}$. Setting $m \approx \sqrt{2^n}$ suffices to make this expectation a constant[26].

So there's a classical algorithm that solves this with $O\left(\sqrt{2^n}\right)$ queries. It's better than querying f everyhere, which would cost 2^n queries. But the square root just amounts to a reduction by a factor of 2 in the exponent. It's still exponentially many queries. What's interesting is that the above is essentially the best possible classical algorithm.

Theorem 18.1. *Any classical algorithm that solves Simon's problem with worst-case success probability at least $\frac{3}{4}$ must make $\Omega\left(\sqrt{2^n}\right)$ queries.*

How do we know this? We *cannot* deduce this simply based in the fact that this is the best algorithm that we could come up with. How can we be sure that there isn't some clever trick that we haven't discovered? Also, notice that f is not an arbitrary 2-to-1 function. It is a 2-to-1 with the Simon property, which is a very special structure. So it's conceivable that a classical algorithm can somehow take advantage of that structure to find a collision in an unusual way.

[26] You may recognize in this analysis the so-called "birthday paradox", where you consider what the chances are that there are two people in a group of (say) 23 people who have the same birthday. It's 50%, assuming that people's birthdays are uniformly distributed.

Theorem 18.1 is true, but it requires a proof. The proof is not as trivial as the argument that two queries are needed for Deutsch's problem, or that 3 queries are needed for the 1-out-of-4 search problem. It requires a more careful argument. If you're interested in seeing the proof, it is in the next subsection 18.2.1. The proof isn't particularly hard, but it requires some set-up. Feel free to skip past subsection 18.2.1 on a first reading.

18.2.1 Proof of classical lower bound for Simon's problem (Theorem 18.1)

In this section, we prove Theorem 18.1. The proof uses some standard techniques that arise in computational complexity; however, this account assumes no prior background in the area.

The first part of the proof is to "play the adversary" by coming up with a way of generating an instance of f that will be hard for any algorithm. Note that picking some *fixed* f will not work very well. A fixed f has a fixed r associated with it and the first two queries of the algorithm *could* be 0^n and r, which would reveal r to the algorithm after only two queries. Rather, we shall *randomly* generate instances of f. First, we pick r at random, uniformly from $\{0,1\}^n - 0^n$. Picking r does not fully specify f but it partitions $\{0,1\}^n$ into 2^{n-1} colliding pairs of the form $\{x, x \oplus r\}$, for which $f(x) = f(x \oplus r)$ will occur. Let us also specify a representative element form each colliding pair, say, the smallest element of $\{x, x \oplus r\}$ in the lexicographic order. Let T be the set of all such representatives: $T = \{s : s = \min\{x, x \oplus r\} \text{ for some } x \in \{0,1\}^n\}$. Then we can define f in terms of a random one-to-one function $\phi : T \to \{0,1\}^n$ uniformly over all the $2^n(2^n-1)(2^n-2)(2^n-3)\cdots(2^n-2^{n-1}+1)$ possibilities. The definition of f can then be taken as

$$f(x) = \begin{cases} \phi(x) & \text{if } x \in T \\ \phi(x \oplus r) & \text{if } x \notin T. \end{cases}$$

We shall prove that no classical probabilistic algorithm can succeed with probability $\frac{3}{4}$ on such instances unless it makes a very large number of queries.

The next part of the proof is to show that, with respect to the above distribution among inputs, we need only consider *deterministic* algorithms (by which we mean ones that make no probabilistic choices). The idea is that any probabilistic algorithm is just a probability distribution over all the deterministic algorithms, so its success probability p is the average of the success probabilities of all the deterministic algorithms (where the average is weighted by the probabilities). At least one deterministic algorithm must have success probability $\geq p$ (otherwise the average would

be less than p). Therefore (because we have a fixed probability distribution of the input instances), we need only consider deterministic algorithms.

Next, consider some deterministic algorithm and the first query that it makes: $(x_1, y_1) \in \{0,1\}^n \times \{0,1\}^n$, where x_1 is the input to the query and y_1 is the output of the query. The result of this will just be a uniformly random element of $\{0,1\}^n$, independent of r. Therefore the first query by itself contains absolutely no information about r.

Now consider the second query (x_2, y_2) (without loss of generality, we can assume that the inputs to all queries are different; otherwise, the redundant queries could be eliminated from the algorithm). There are two possibilities: $x_1 \oplus x_2 = r$ (collision) or $x_1 \oplus x_2 \neq r$ (no collision). In the first case, we will have $y_1 = y_2$ and so the algorithm can deduce that $r = x_1 \oplus x_2$. But the first case arises with probability only $\frac{1}{2^n - 1}$. With probability $1 - \frac{1}{2^n - 1}$, we are in the second case, and all that the algorithm deduces about r is that $r \neq x_1 \oplus x_2$ (it has ruled out just one possibility among $2^n - 1$).

We continue our analysis of the process by induction on the number of queries. Suppose that $k-1$ queries, $(x_1, y_1), \ldots, (x_{k-1}, y_{k-1})$ have been made without any collisions so far (i.e., for all $1 \leq i < j \leq k-1$, $y_i \neq y_j$). Then all that has been deduced about r is that $r \neq x_i \oplus x_j$, for all $1 \leq i < j \leq k-1$. Therefore, at most $\binom{k-1}{2} = (k-1)(k-2)/2$ possibilities for r have been eliminated (up to one value of r is eliminated for each pair of x_i and x_j). Since there are $2^n - 1$ values of r to begin with (recall that $r \neq 0^n$), it follows that there are at least $2^n - 1 - (k-1)(k-2)/2$ possibilities of r that have not yet been eliminated. When the next query (x_k, y_k) is made, the number of potential collisions arising from it are at most $k-1$ (the number of previous queries). Therefore, the probability of a collision at query k is at most

$$\frac{k-1}{2^n - 1 - (k-1)(k-2)/2} \leq \frac{2k}{2^{n+1} - k^2}. \tag{157}$$

Let's review the expression on the left side of Eq. (157). For the denominator, there are at least $2^n - 1 - (k-1)(k-2)/2$ possibilities of r remaining at the point of query k (assuming that no collisions have occurred yet). And, for the numerator, among those remaining values of r, there are $k-1$ values of r that cause a collision to occur for query x_k, namely the values in the set $\{x_k \oplus x_1,\ x_k \oplus x_2,\ \ldots,\ x_k \oplus x_{k-1}\}$.

Since the collision probability bound in Eq. (157) holds all k, the probability of a collision occurring somewhere among m queries is at most the sum of the right side of Eq. (157) with k varying from 1 to m:

$$\sum_{k=1}^{m} \frac{2k}{2^{n+1} - k^2} \leq \sum_{k=1}^{m} \frac{2m}{2^{n+1} - m^2} \leq \frac{2m^2}{2^{n+1} - m^2}. \tag{158}$$

If this quantity is to be at least $\frac{3}{4}$ then

$$\frac{2m^2}{2^{n+1} - m^2} \geq \frac{3}{4}. \tag{159}$$

It is an easy exercise to solve for m in the above inequality, yielding $m \geq \sqrt{(6/11)2^n}$, which gives the desired bound.

Actually, there is a slight technicality remaining. We have shown that $\sqrt{(6/11)2^n}$ queries are necessary *to attain a collision* with probability $\frac{3}{4}$; whereas the algorithm is not technically required to make queries that include a collision. The algorithm is only required to deduce r, and it's conceivable that an algorithm could deduce r some other way without a collision occurring. But any algorithm that deduces r can be modified so that it makes one additional query that collides with a previous one. So we have a slightly smaller lower bound of $\sqrt{(6/11)2^n} - 1$, but this is still $\Omega(\sqrt{2^n})$.

18.3 Quantum algorithm for Simon's problem

Before showing you the algorithm for Simon's problem, I'd like to show you a particularly useful way of viewing the multi-qubit Hadamard transform $H \otimes H \otimes \cdots \otimes H$ and the structure of $\{0,1\}^n$.

18.3.1 Understanding $H \otimes H \otimes \cdots \otimes H$

To start with, let's look at how $H \otimes H \otimes \cdots \otimes H = H^{\otimes n}$ (a Hadamard transform on each of n qubits) affects the computational basis states.

First, for the state $|00\ldots0\rangle = |0^n\rangle$,

$$H^{\otimes n} |0^n\rangle = \tfrac{1}{\sqrt{2^n}} \sum_{b \in \{0,1\}^n} |b\rangle . \tag{160}$$

It turns out that there's this nice expression for applying $H^{\otimes n}$ to any computational basis state $|a\rangle$ ($a \in \{0,1\}^n$). It's a uniform superposition of the computational basis states, but with certain phases.

Theorem 18.2. *For all $a \in \{0,1\}^n$,*

$$H^{\otimes n} |a\rangle = \tfrac{1}{\sqrt{2^n}} \sum_{b \in \{0,1\}^n} (-1)^{a \cdot b} |b\rangle , \tag{161}$$

where $a \cdot b = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \bmod 2$.

For example,

$$H \otimes H = \frac{1}{\sqrt{4}} \begin{array}{c} \begin{matrix} 00 & 01 & 10 & 11 \end{matrix} \\ \begin{bmatrix} +1 & +1 & +1 & +1 \\ +1 & -1 & +1 & -1 \\ +1 & +1 & -1 & -1 \\ +1 & -1 & -1 & +1 \end{bmatrix} \end{array} \begin{matrix} 00 \\ 01 \\ 10 \\ 11 \end{matrix} \tag{162}$$

Note that, for each $a, b \in \{0,1\}^2$, the sign of entry (a, b) is $(-1)^{a \cdot b}$.

Proof of Theorem 18.2. The proof is this simple calculation

$$\begin{aligned} H^{\otimes n} |a\rangle &= \left(H |a_1\rangle\right) \otimes \cdots \otimes \left(H |a_n\rangle\right) && (163) \\ &= \left(\tfrac{1}{\sqrt{2}} |0\rangle + \tfrac{1}{\sqrt{2}}(-1)^{a_1} |1\rangle\right) \otimes \cdots \otimes \left(\tfrac{1}{\sqrt{2}} |0\rangle + \tfrac{1}{\sqrt{2}}(-1)^{a_n} |1\rangle\right) && (164) \\ &= \left(\tfrac{1}{\sqrt{2}} \sum_{b_1 \in \{0,1\}} (-1)^{a_1 b_1} |b_1\rangle\right) \otimes \cdots \otimes \left(\tfrac{1}{\sqrt{2}} \sum_{b_n \in \{0,1\}} (-1)^{a_n b_n} |b_n\rangle\right) && (165) \\ &= \tfrac{1}{\sqrt{2^n}} \sum_{b \in \{0,1\}^n} (-1)^{a_1 b_1} \cdots (-1)^{a_n b_n} |b\rangle && (166) \\ &= \tfrac{1}{\sqrt{2^n}} \sum_{b \in \{0,1\}^n} (-1)^{a_1 b_1 + \cdots + a_n b_n} |b\rangle . && (167) \end{aligned}$$

■

So we have a nice expression for applying $H^{\otimes n}$ on computational basis states. In particular, notice how an expression that looks like a dot-product of two n-bit strings (in modulo 2 arithmetic) arises.

18.3.2 Viewing $\{0,1\}^n$ as a discrete vector space

Now let's think about the set $\{0,1\}^n$. It's often useful to associate these strings with mathematical structures, such as the integers $\{0, 1, 2, \ldots, 2^n - 1\}$.

But we can also think of the set $\{0,1\}^n$ as an n-dimensional vector space, where the components of each vector are 0 and 1, and the arithmetic is modulo 2 (which is equivalent to using $\oplus$ for addition and $\wedge$ for multiplication). This is different from a vector space over the field $\mathbb{R}$ or $\mathbb{C}$. But the set $\{0,1\}$, with addition and multiplication modulo 2 (which we'll denoted as $\mathbb{Z}_2$), is a *field*, meaning that it shares some key structural properties that the real and complex numbers have (I won't go

into the details of these properties here). It's perfectly valid to have a vector space over a finite field like $\mathbb{Z}_2$. The linear algebra notions of *subspace*, *dimension* and *linear independence*, make perfect sense over such vector spaces.

And this brings us to that dot-product expression that arose in the n-fold tensor product of the Hadamard. For vector spaces over $\mathbb{R}$ and $\mathbb{C}$, the dot-product[27] is an *inner product*, and has useful properties. Two vectors are *orthogonal* if and only if their inner product is zero.

Technically, the dot-product in our finite field $\mathbb{Z}_2$ scenario is *not* an inner product. An inner product has the property that, for any vector v, it holds that $v \cdot v = 0$ if and only if $v = 0$ (the zero vector). In our finite field vector space, there are non-zero binary strings whose inner products with themselves are 0. Can you think of one? Any binary string with an even number of 1s has dot product 0 with itself.

Nevertheless, this dot product does have *some* nice properties. For example, the space can be decomposed into "orthogonal" subspaces whose dimensions add up to n. Two spaces are deemed "orthogonal" if every point in one has dot-product 0 with every point in the other. Here is a schematic picture of a decomposition of $\{0,1\}^3$ decomposed into a 1-dimensional space and an orthogonal 2-dimensional space.

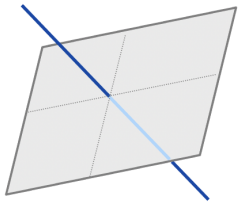

Figure 130: Schematic picture of $\{0,1\}^3$ decomposed into two orthogonal subspaces.

Of course the picture is really for vector spaces over the field $\mathbb{R}$, not the finite field $\mathbb{Z}_2$. So it should just be seen as an intuitive guide, rather than a literal depiction.

A word of caution: for n-qubit systems, there are two spaces in play. One space is the n-dimensional discrete vector space $\{0,1\}^n$ which is over the finite field $\mathbb{Z}_2$. This is associated with the *labels* of the computational basis states. The other space is the 2^n-dimensional space over $\mathbb{C}$, spanned by the 2^n computational basis states (technically, a *Hilbert space*). Do not conflate these different spaces! For example, $00\ldots0$ is the zero vector in the finite vector space. But $|00\ldots0\rangle$ lives in the Hilbert space, and it's definitely not the zero vector. It's a quantum state, which is a vector of length one.

[27] In the case of field $\mathbb{C}$ we need to take complex conjugates one of the vectors in the dot product.

18.3.3 Simon's algorithm

Applying definition 16.1, the f-queries look like this.

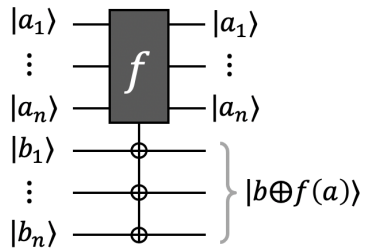

Figure 131: f-query for Simon's Problem.

The f-query acts on $2n$ qubits and, in the computational basis, f of the first n bits is XORed onto the last n bits. With queries like this, it's not so clear how we can "query in the phase", as we did for all the quantum algorithms in section 17.

We'll start out differently, with all the n target qubits just in state $|0\rangle$. But we will put the inputs to f into a uniform superposition of all the n-bit strings. And then we'll perform an f-query.

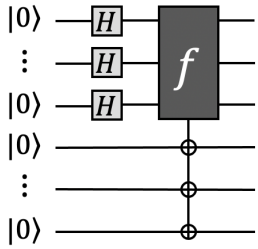

Figure 132: Beginning of Simon's algorithm.

What's the output state of this circuit? The state right after the query is

$$\frac{1}{\sqrt{2^n}} \sum_{a \in \{0,1\}^n} |a\rangle \, |f(a)\rangle \, . \tag{168}$$

Let's consider this state. It's sometimes said that what makes quantum computers so powerful is that they can actually perform several computations at the same time. But this view is misleading. The state was obtained by making just one single query, and it appears to contain information about all of the values of the function f. However, it's not possible to extract more than one value of the function from this state. In particular, if we measure this state in the computational basis then what we end up

with is just one pair $(a, f(a))$, where a is sampled from the uniform distribution. And you don't need any quantum devices to generate such a sample. You could just flip a coin n times to create a random a and then perform one query to get $f(a)$.

So how *should* we think about quantum algorithms? With a state like that in Eq. (168), rather than measuring in the computational basis, we can instead—via a unitary operation—measure with respect to *another* basis. In some cases, for a well-chosen basis, we can acquire information about some *global property* of f (*instead of* the value of f at some specific point).

Let's try to find a useful measurement basis for the case where f satisfies the Simon property. The inputs to the function partition into colliding pairs. It helps to look at our example in figure 129 again for reference. It's reproduced here for convenience.

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

Figure 133: Copy of figure 129: A function satisfying the Simon property.

Let's define a set T so as to consist of one element from each colliding pair. In the example, we take one of the two green points, one of the two blue points, one of the two purple points and one of the two red points. Which one we choose won't matter; what's important is that there exists such a set T. In the example, we could set $T = \{000, 100, 010, 011\}$.

Notice that, for each element $a \in T$, the *other* element of the colliding pair is $a \oplus r$. We don't know what r is (that's what we're trying to find by the algorithm). But we know that f satisfies the Simon property and that *there exists* an associated r. (In the example, $r = 101$.)

Therefore, if we combine the elements of T with the elements of the set $T \oplus r$ then

we get all of $\{0,1\}^n$. This enables us to rewrite the state in Eq. (168) as

$$\tfrac{1}{\sqrt{2^n}} \sum_{a \in T} \Big(|a\rangle \, |f(a)\rangle + |a \oplus r\rangle \, |f(a \oplus r)\rangle \Big), \tag{169}$$

where we are only summing over the elements of T, but, for each $a \in T$, we include two terms: one for a and one for $a \oplus r$.

Now, since f satisfies the Simon property with the associated r, for each a, we have that $f(a) = f(a \oplus r)$. That's exactly what the Simon property says. So we can write the expression in Eq. (169) as

$$\tfrac{1}{\sqrt{2^n}} \sum_{a \in T} \Big(|a\rangle + |a \oplus r\rangle \Big) \, |f(a)\rangle \, . \tag{170}$$

Suppose that we now measure the last n qubits in the computational basis. Then the state of the last n qubits collapses to some value $f(a)$ and the residual state of the first n qubits is a uniform superposition of the pre-images of that value. That is, the state of the first n qubits becomes

$$\tfrac{1}{\sqrt{2}} \, |a\rangle + \tfrac{1}{\sqrt{2}} \, |a \oplus r\rangle \, . \tag{171}$$

for a random $a \in \{0,1\}^n$. What can we do with this state? If we could somehow extract both a and $a \oplus r$ from this state then we could deduce the value of r (by taking their XOR, $a \oplus (a \oplus r) = r$). But we can only measure the state once, after which its state collapses.

If we measure in the computational basis, then we just get either a or $a \oplus r$, neither of which is sufficient to learn anything about the value of r. We can think of measuring in the computational basis this way: we first randomly choose a color (that's what happens when we measure the last n qubits), and then we randomly choose one of the two elements of that color (that's what happens when we measure the first n qubits). So the net effect of all this is to get just a random n-bit string, which is devoid of any information about the structure of f. So, if we want make progress than we should definitely *not* measure the first n qubits in the computational basis.

Something quite remarkable happens if we apply a Hadamard transform to each of the n qubits before measuring in the computational basis. We can calculate the

state resulting from the Hadamard transform using Theorem 18.2 as

$$H^{\otimes n}\Big(\tfrac{1}{\sqrt{2}}\,|a\rangle + \tfrac{1}{\sqrt{2}}\,|a\oplus r\rangle\Big) = \tfrac{1}{\sqrt{2^{n+1}}}\left(\sum_{b\in\{0,1\}^n}(-1)^{a\cdot b}\,|b\rangle + \sum_{b\in\{0,1\}^n}(-1)^{(a\oplus r)\cdot b}\,|b\rangle\right) \tag{172}$$

$$= \tfrac{1}{\sqrt{2^{n+1}}}\left(\sum_{b\in\{0,1\}^n}(-1)^{a\cdot b}\,|b\rangle + \sum_{b\in\{0,1\}^n}(-1)^{a\cdot b}(-1)^{r\cdot b}\,|b\rangle\right) \tag{173}$$

$$= \tfrac{1}{\sqrt{2^{n+1}}}\left(\sum_{b\in\{0,1\}^n}(-1)^{a\cdot b}\Big(1+(-1)^{r\cdot b}\Big)\,|b\rangle\right). \tag{174}$$

Why is this interesting? Let's think about what happens if we measure *this* state in the computational basis. Notice that, for each $b \in \{0,1\}^n$,

$$\Big(1+(-1)^{r\cdot b}\Big) = \begin{cases} 2 & \text{if } r\cdot b = 0 \\ 0 & \text{if } r\cdot b = 1. \end{cases} \tag{175}$$

Therefore, the probability of each $b \in \{0,1\}^n$ occurring as the outcome is

$$\begin{cases} \frac{1}{2^{n-1}} & \text{if } r\cdot b = 0 \\ 0 & \text{if } r\cdot b = 1. \end{cases} \tag{176}$$

In other words, the outcome of the measurement is a uniformly distributed random b in the orthogonal complement of r (that is, for which $r\cdot b = 0$).

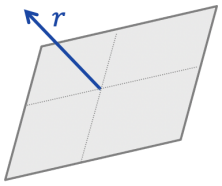

Figure 134: Schematic illustration of $r \in \{0,1\}^n$ and the set $\{b \in \{0,1\}^n : r\cdot b = 0\}$.

This b does not produce enough information for us to deduce r, but it reveals *partial information* about r. Namely that the bits of r satisfy the linear equation

$$b_1 r_1 + b_2 r_2 + \cdots + b_n r_n \equiv 0 \pmod 2. \tag{177}$$

And we can acquire more information about r by repeating the procedure.

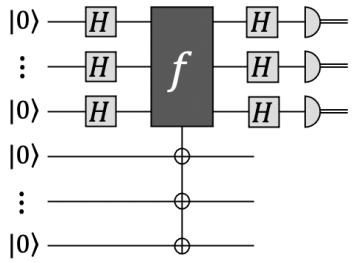

Figure 135: Each execution of this procedure yields a random $b \in \{0,1\}^n$ such that $r \cdot b = 0$.

Each execution of the procedure in figure 135 produces[28] an independent random $b \in \{0,1\}^n$ that is orthogonal to r (in the sense that $r \cdot b = 0$). Suppose that we repeat the process $n-1$ times (so the number of f-queries is $n-1$). Then, combining the resulting b's, we obtain a system of $n-1$ linear equations (in mod 2 arithmetic)

$$\begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,n} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n-1,1} & b_{n-1,2} & \cdots & b_{n-1,n} \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{178}$$

If the $n-1 \times n$ matrix has rank $n-1$ then there is a unique nonzero r solution to the system, which can be easily found by Guassian elimination. What's the probability that this matrix has full rank? It turns out that this probability is a constant independent of n.

Lemma 18.1. *The matrix in Eq. (178), whose rows consist of $n-1$ independent random samples from the uniform distribution on a linear space of dimension $n-1$, has full rank with probability at least $\frac{1}{4}$.*

Proof. Let's start by considering the first row of the matrix in Eq. (178). This single vector is linearly independent if and only if it's non-zero, which occurs with probability $1 - \frac{1}{2^{n-1}}$ (there is one bad case among 2^{n-1} possibilities). Next, consider the first *two* rows. Conditional on the first row being linearly independent, the first two rows are linearly independent if and only if the second row is non-zero and not equal to the first row. This conditional probability is $1 - \frac{2}{2^{n-1}} = 1 - \frac{1}{2^{n-2}}$ (there are two bad cases among 2^{n-1}). Similarly, conditional on the first two rows being linearly independent, the first three rows are linearly independent if and only if the third row is not in the

[28] The outcome of the measurement of the first n qubits is the same whether or not the last n qubits are measured.

span of the first two rows. This occurs with probability $1 - \frac{4}{2^{n-1}} = 1 - \frac{1}{2^{n-3}}$ (there are four bad cases). Continuing in this manner, conditional on the first $k-1$ rows being linearly independent, the first k rows are linearly independent if and only if the k-th row is not a linear combination of the first $k-1$ rows. And this occurs with probability $1 - \frac{2^{k-1}}{2^{n-1}} = 1 - \frac{1}{2^{n-k}}$ (there are 2^{k-1} bad cases). Multiplying all these conditional probabilities for rows 1 through $n-1$, we obtain

$$\left(1 - \frac{1}{2}\right)\left(1 - \frac{1}{2^2}\right) \cdots \left(1 - \frac{1}{2^{n-2}}\right)\left(1 - \frac{1}{2^{n-1}}\right) \tag{179}$$

as the probability that the matrix in Eq. (178) is full rank. That this product is at least $\frac{1}{4}$ is left as an exercise. ■

Exercise 18.1. Show that the expression in Eq. (179) is at least $\frac{1}{4}$.

So we now have a quantum algorithm that makes $n-1$ queries in all and succeeds in finding r with probability at least $\frac{1}{4}$. This fails with probability at most $\frac{3}{4}$. It's easy to reduce the failure probability to $(\frac{3}{4})^5 < \frac{1}{4}$ by repeating the entire procedure five times.

In conclusion, $\Omega(\sqrt{2^n})$ queries are necessary for any classical algorithm to attain success probability $\frac{3}{4}$, whereas order $O(n)$ queries are sufficient for a quantum algorithm to attain success probability $\frac{3}{4}$.

18.4 Significance of Simon's problem

Now, what should we make of Simon's problem and Simon's algorithm?

It's a black-box problem that was specially designed to be very hard for probabilistic classical algorithms and easy for quantum algorithms—thereby improving on previous classical vs quantum query cost separations.

problem	quantum	classical deterministic	classical probabilistic
Deutsch	1	2	2
1-out-of-4 search	1	3	3
Const. vs. balanced	1	$2^{n-1}+1$	$O(1)$
Simon	$O(n)$	$\Omega(2^{n/2})$	$\Omega(2^{n/2})$

Figure 136: Summary of query costs for problems considered so far.

But Simon's problem doesn't immediately look like a problem that one would care about in the real world. When Simon's work first came out, people were wondering what to make of it. Although it provided a very strong classical vs. quantum query cost separation, it looked like a contrived problem. Moreover, a contrived *black-box* problem, which is not even a conventional computing problem, involving input data.

This may be one's first impression, but there's actually more to it than that. Look again at the Simon property: for all a, $f(a) = f(a \oplus r)$. This is kind of like a periodicity property of a function, which would be written as: for all a, $f(a) = f(a + r)$. And periodic functions arise naturally in many contexts. We'll soon see that variations of Simon's problem in this direction are very fruitful.

19 The Fourier transform

Up until now, the Hadamard transform $H = \frac{1}{\sqrt{2}}\left[\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right]$ (acting on qubits) has played a prominent role in quantum algorithms. There is a natural generalization of H to m-dimensional systems, called the *Fourier transform*, which is also very useful for quantum algorithms.

19.1 Definition of the Fourier transform modulo m

We begin by considering complex numbers the form

$$\omega = e^{2\pi i/m}, \tag{180}$$

which we refer to as *(primitive) m^{th} roots of unity*. Clearly, $\omega^m = 1$. Here's where ω, and all its powers, lie in the complex plane $\mathbb{C}$.

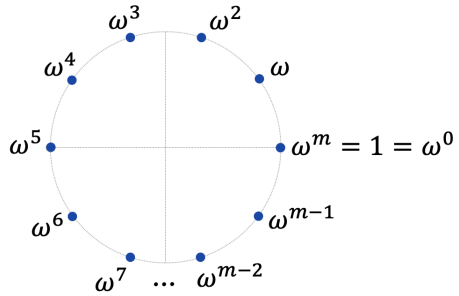

Figure 137: The powers of ω are m equally spaces points on the unit circle in $\mathbb{C}$.

If we sum all these powers of ω, we get zero: $1 + \omega + \omega^2 + \cdots + \omega^{m-1} = 0$. Can you see why? Moreover, if we sum all the powers of ω^k (assuming $1 \leq k \leq m-1$) we also get zero.

Exercise 19.1 (Hint: use formula for sum of geometric sequence)**.** Prove that, for all $k \in \{1, 2, \ldots, m-1\}$, it holds that

$$1 + \omega^k + \omega^{2k} + \cdots + \omega^{(m-1)k} = \sum_{j=0}^{m-1} \omega^{jk} = 0. \tag{181}$$

What about the powers of ω^m? Since $\omega^m = 1$, it's obvious that $\sum_{j=0}^{m-1} \omega^{jm} = m$.

Now we can define the Fourier transform modulo m.

Definition 19.1 (Fourier transform)**.** The *Fourier transform modulo* m is the $m \times m$ matrix

$$F_m = \frac{1}{\sqrt{m}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{m-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(m-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{m-1} & \omega^{2(m-1)} & \cdots & \omega^{(m-1)^2} \end{bmatrix}. \tag{182}$$

Note that (after the normalization factor $\frac{1}{\sqrt{m}}$) the first column is 1s, the second column is the powers of ω, the third column is powers of ω^2 and so on.

Exercise 19.2. Prove that F_m is unitary.

Exercise 19.3. Prove that the inverse Fourier transform F_m^* is

$$F_m^* = \frac{1}{\sqrt{m}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \cdots & \omega^{-(m-1)} \\ 1 & \omega^{-2} & \omega^{-4} & \cdots & \omega^{-2(m-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{-(m-1)} & \omega^{-2(m-1)} & \cdots & \omega^{-(m-1)^2} \end{bmatrix}. \tag{183}$$

For all $a \in \{0, 1, \ldots, m-1\}$, applying the Fourier transform F_m to the computational basis state $|a\rangle$ results in the state

$$F_m |a\rangle = \tfrac{1}{\sqrt{m}} \sum_{j=0}^{m-1} \omega^{ja} |j\rangle, \tag{184}$$

which corresponds to column a of F_m. We call this a *Fourier basis state*. Similarly, applying the inverse Fourier transform F_m^* to $|a\rangle$ results in the state

$$F_m^* |a\rangle = \tfrac{1}{\sqrt{m}} \sum_{j=0}^{m-1} \omega^{-ja} |j\rangle. \tag{185}$$

If there are n registers that are each m-dimensional, and you apply F_m to each register (which is $F_m \otimes F_m \otimes \cdots \otimes F_m = F_m^{\otimes n}$ on the n-register system) then, for all $(a_1, a_2, \ldots, a_n) \in \mathbb{Z}_m^n$,

$$(F_m)^{\otimes n} |a_1, a_2, \ldots, a_n\rangle = \frac{1}{\sqrt{m^n}} \sum_{b \in \mathbb{Z}_m^n} \omega^{a \cdot b} |b_1, b_2, \ldots, b_n\rangle \tag{186}$$

$$(F_m^*)^{\otimes n} |a_1, a_2, \ldots, a_n\rangle = \frac{1}{\sqrt{m^n}} \sum_{b \in \mathbb{Z}_m^n} \omega^{-a \cdot b} |b_1, b_2, \ldots, b_n\rangle, \tag{187}$$

where $a \cdot b = (a_1, a_2, \ldots, a_n) \cdot (b_1, b_2, \ldots, b_n) = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n \bmod m$. Note that Eqns. (186)(187) generalize Theorem 18.2 in section 18.3.1 from the modulo 2 case to the modulo m case.

19.2 A very simple application of the Fourier transform

We will soon see, in section 21, that it's possible to efficiently solve the celebrated *discrete log problem* (which will be defined in section 20) by reducing it to a generalization of Simon's problem (section 21.2), where the modulus is m (as opposed to 2). This generalization can be solved along the lines of Simon's algorithm—using Fourier transforms in place of Hadamard transforms.

In this section, we consider a very simple query problem where a quantum algorithm that employs the Fourier transform outperforms what any classical query algorithm can do. The reduction in query cost is modest: the quantum algorithm makes one f-query; whereas any classical algorithm must make two f-queries. Although this is not a dramatic efficiency improvement by a quantum algorithm, it shows the Fourier transform in action in a simple setting, where some of its interesting properties are easy to observe and analyze.

Definition 19.2 (Linear coefficient problem)**.** Let $m \geq 2$. Let $a, b \in \mathbb{Z}_m$ and define $f : \mathbb{Z}_m \to \mathbb{Z}_m$ as

$$f(x) = ax + b \bmod m, \tag{188}$$

for all $x \in \mathbb{Z}_m$. Assume that you are given access to a black-box for the function f, but you are given no other information about f. You don't know what the linear coefficient a is, nor the additive constant b. Your goal is to find the value of the linear coefficient a.

It's not hard to see that, in the special case where $m = 2$, this is equivalent to Deutsch's problem (section 17.1). In that case, the four functions are all of the form of Eq. (188) and $f(0) = f(1)$ if and only if $a = 0$. Moreover, it is straightforward to show that, for all $m \geq 2$, any classical algorithm for the linear coefficient problem problem must make at least two f-queries.

Exercise 19.4. Show that, for classical algorithms, *two* f-queries are necessary and sufficient to solve the linear coefficient problem.

Next, we will see a quantum algorithm that solves this problem with only one quantum f-query.

We need to define the notion of a *quantum* f-query for functions of the form $f : \mathbb{Z}_m \to \mathbb{Z}_m$. A reasonable definition is as the unitary operation that maps each basis state $|x\rangle\,|y\rangle$ to

$$|x\rangle\,|y+f(x) \bmod m\rangle\,, \tag{189}$$

for all $x, y \in \mathbb{Z}_m$. Here is notation for a quantum f-query (which makes sense for *any* $f : \mathbb{Z}_m \to \mathbb{Z}_m$).

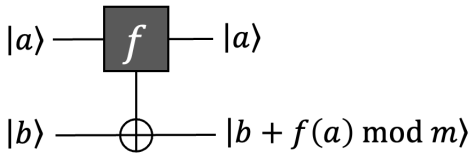

Figure 138: Quantum f-query (for $f : \mathbb{Z}_m \to \mathbb{Z}_m$).

Note the similarity with figure 112 (for the case where $f : \{0,1\} \to \{0,1\}$).

Given the resemblance of the leading coefficient problem to Deutsch's problem, we can draw inspiration from the quantum algorithm for that problem (section 17.1), substituting F_m and F_m^* for Hadamard transforms. In fact, our quantum algorithm will be the following quantum circuit.

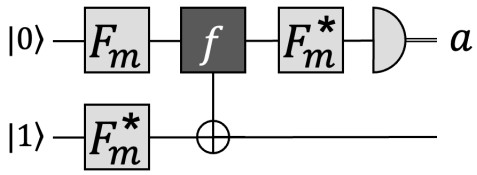

Figure 139: Quantum algorithm for the linear coefficient problem.

Note the resemblance to the quantum circuit in figure 115; when $m = 2$, the two quantum circuits are identical. Why is the measured output of this quantum circuit the linear coefficient a? We will carefully go through then role that each of the Fourier transforms plays in the computation.

In figure 115, the Hadamard transform on the second register creates the special state $|-\rangle$ in the target register of the f-query, so that f-queries have the effect of

querying in the phase: $|x\rangle \mapsto (-1)^{f(x)} |x\rangle$ (for $x \in \{0,1\}$). An analogue of querying in the phase for $\mathbb{Z}_m$-valued registers is to implement a mapping of the form

$$|x\rangle \mapsto \omega^{f(x)} |x\rangle \tag{190}$$

(for $x \in \mathbb{Z}_m$), where $\omega = e^{2\pi i/m}$. This is achieved by setting the second register to the state

$$|\psi\rangle = F_m^* |1\rangle \tag{191}$$

$$= \frac{1}{\sqrt{m}} \sum_{b=0}^{m-1} \omega^{-b} |b \bmod m\rangle . \tag{192}$$

To see why this causes f-queries to implement the mapping in Eq. (190), first define a mod m generalization of the unitary operation X as the m-dimensional unitary operation that, in the computational basis, adds 1 modulo m to a register. Let's call this the *increment modulo m* operation and denote it as X_m. For all $a \in \mathbb{Z}_m$,

$$X_m |a\rangle = |a + 1 \bmod m\rangle . \tag{193}$$

The effect of X_m on $|\psi\rangle$ (the state defined in Eq. (191)) is

$$X_m |\psi\rangle = X_m \left(\tfrac{1}{\sqrt{m}} \sum_{b=0}^{m-1} \omega^{-b} |b\rangle \right) \tag{194}$$

$$= \tfrac{1}{\sqrt{m}} \sum_{b=0}^{m-1} \omega^{-b} |b + 1 \bmod m\rangle \tag{195}$$

$$= \tfrac{1}{\sqrt{m}} \sum_{c=0}^{m-1} \omega^{-(c-1)} |c \bmod m\rangle \tag{196}$$

$$= \tfrac{1}{\sqrt{m}} \sum_{c=0}^{m-1} \omega\omega^{-c} |c \bmod m\rangle \tag{197}$$

$$= \omega |\psi\rangle . \tag{198}$$

More generally, we have $(X_m)^k |\psi\rangle = \omega^k |\psi\rangle$ (in other words, adding k modulo m to state $|\psi\rangle$ results in state $\omega^k |\psi\rangle$). From this it follows that an f-query applied to state $|x\rangle |\psi\rangle$ produces the state $\omega^{f(x)} |x\rangle |\psi\rangle$, thereby implementing a mapping of the form in Eq. (190) (with respect to the first register).

Following the outline of Deutsch's algorithm (section 17.1) as a guide, the Fourier

transform on the first qubit creates the superposition

$$F_m |0\rangle = \tfrac{1}{\sqrt{m}}\big(|0\rangle + |1\rangle + \cdots + |m-1\rangle\big) \tag{199}$$

$$= \tfrac{1}{\sqrt{m}} \sum_{x=0}^{m-1} |x\rangle \,. \tag{200}$$

Applying an f-query to this state (with the second register set to $|\psi\rangle$), results in the state

$$\tfrac{1}{\sqrt{m}} \sum_{x=0}^{m-1} \omega^{f(x)} |x\rangle = \tfrac{1}{\sqrt{m}} \sum_{x=0}^{m-1} \omega^{ax+b} |x\rangle \tag{201}$$

$$= \omega^b \left(\tfrac{1}{\sqrt{m}} \sum_{x=0}^{m-1} \omega^{ax} |x\rangle \right) \tag{202}$$

$$= \omega^b F_m |a\rangle \,. \tag{203}$$

Since this state is $F_m |a\rangle$ (with a global phase of ω^b), applying F_m^* to the first register after the query results in the state $\omega^b |a\rangle$. Applying a measurement to this first register results in a, as required. Therefore, the quantum circuit in figure 139 does indeed solve the linear coefficient problem.

19.3 Computing the Fourier transform modulo 2^n

The Fourier transform F_m is defined in section 19.1. We're interested in computing F_m efficiently by a quantum circuit consisting of elementary operations acting on qubits. There's an elegant way to do this in the case where $m = 2^n$. Note that F_{2^n} is a unitary operation acting on n qubits. I will show you a fairly simple quantum circuit consisting of $O(n^2)$ gates that computes F_{2^n}.

It's useful to visualize how the powers of a (primitive) 2^n-th root of unity ω are aligned in the complex plane. They are 2^n equally spaced points on the unit circle, and here's what they look like when $n = 3$.

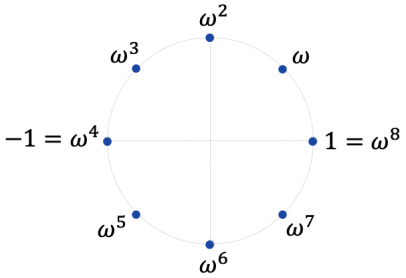

Figure 140: Powers of ω, a primitive 8-th root of unity in $\mathbb{C}$.

For a 2^n-th root of unity ω, a couple of simple but valuable observations are:

1. $\omega^{2^{n-1}} = -1$. (For example, in figure 140, ω is an 8-th root of unity and $\omega^4 = -1$.)
2. ω^2 is a 2^{n-1}-th root of unity. (In figure 140, ω^2 is an 4-th root of unity.)

19.3.1 Expressing F_{2^n} in terms of $F_{2^{n-1}}$

To get a feeling for how this works, let's first consider the case where $n = 3$. For each $a \in \{0,1\}^3 \equiv \{0,1,2,3,4,5,6,7\}$, the corresponding Fourier basis state $F_8 |a\rangle$ is

$$|000\rangle + \omega^a|001\rangle + \omega^{2a}|010\rangle + \omega^{3a}|011\rangle + \omega^{4a}|100\rangle + \omega^{5a}|101\rangle + \omega^{6a}|110\rangle + \omega^{7a}|111\rangle\,.$$

(where the normalization factor $\frac{1}{\sqrt{8}}$ is omitted). Question: Are these three qubits entangled or in a product state? In fact, this state can be written as

$$\begin{aligned} |0\rangle \otimes \Big(|00\rangle + \omega^a|01\rangle + &\omega^{2a}|10\rangle + \omega^{3a}|11\rangle\Big) \\ &+ \omega^{4a}|1\rangle \otimes \Big(|00\rangle + \omega^a|01\rangle + \omega^{2a}|10\rangle + \omega^{3a}|11\rangle\Big) \qquad (204) \\ = \Big(|0\rangle + \omega^{4a}|1\rangle\Big) &\otimes \Big(|00\rangle + \omega^a|01\rangle + \omega^{2a}|10\rangle + \omega^{3a}|11\rangle\Big) \qquad (205) \\ = \Big(|0\rangle + \omega^{4a}|1\rangle\Big) &\otimes \Big(|0\rangle + \omega^{2a}|1\rangle\Big) \otimes \Big(|0\rangle + \omega^a|1\rangle\Big). \qquad (206) \end{aligned}$$

So $F_8 |a\rangle$ is a product of three 1-qubit states.

This generalizes to $F_{2^n} |a\rangle$, for all $n \geq 1$, as follows.

Lemma 19.1. *For all $n \geq 1$ and $a \in \{0,1\}^n$,*

$$F_{2^n}|a\rangle = \Big(|0\rangle + \omega^{2^{n-1}a}|1\rangle\Big) \otimes \cdots \otimes \Big(|0\rangle + \omega^{4a}|1\rangle\Big) \otimes \Big(|0\rangle + \omega^{2a}|1\rangle\Big) \otimes \Big(|0\rangle + \omega^a|1\rangle\Big), \qquad (207)$$

where there is a normalization factor of $\frac{1}{2^{n/2}}$.

Our quantum circuit for computing the Fourier transform will make use of this structure. Remember that, if we compute a linear operator that matches the Fourier transform on all computational basis states then it must be the Fourier transform.

Let's return our attention to the $n = 3$ case, where the Fourier basis states are

$$F_8 |a\rangle = \Big(|0\rangle + \omega^{4a}|1\rangle\Big) \otimes \Big(|0\rangle + \omega^{2a}|1\rangle\Big) \otimes \Big(|0\rangle + \omega^a|1\rangle\Big), \qquad (208)$$

for all $a \in \{0,1\}^3$, where ω is a primitive 8-th root of unity. Each $a = a_2 a_1 a_0$ denotes an element of $\{0,1,2,3,4,5,6,7\}$ in the usual way (a_0 is the low-order bit and a_2 is the high-order bit).

a	$a_2a_1a_0$
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Figure 141: Binary representation of numbers in $\{0, 1, 2, 3, 4, 5, 6, 7\}$.

Consider the state of the first qubit in Eq. (208). Since $\omega^4 = -1$, the state of the first qubit simplifies to $|0\rangle + (-1)^a |1\rangle$. Note that this is $|+\rangle$ when a is even and $|-\rangle$ when a is odd. The parity of a is determined by its low-order bit a_0. Therefore the first qubit of $F_8 |a\rangle$ is simply $H |a_0\rangle$.

What about the remaining qubits? Let $|\psi\rangle$ denote the second and third qubit in Eq. (208). That is,

$$|\psi\rangle = \Big(|0\rangle + \omega^{2a}|1\rangle\Big) \otimes \Big(|0\rangle + \omega^{a}|1\rangle\Big). \tag{209}$$

Now, please look at the $n = 2$ case of the expression in Eq. (207) in Lemma 19.1. Does $|\psi\rangle$ look like $F_4 |a\rangle$?

There is certainly a superficial resemblance; however, $|\psi\rangle$ and $F_4 |a\rangle$ are not *exactly* the same. The state $F_4 |a\rangle$ is with respect to a 4-th root of unity—not an 8-th root of unity. Another difference is that F_4 acts on 2 qubits; whereas the parameter a in Eq. (209) is a 3-bit integer.

It will be fruitful to explore in more detail the difference between $|\psi\rangle$ and $F_4 |a\rangle$. Let's see what these states look like in terms of the digits $a_2a_1a_0$ of the binary representation of a. We'll use the Greek letter ϖ to denote the 4-th root of unity ($\varpi = e^{2\pi i/4}$), while reserving ω for the 8-th root of unity ($\omega = e^{2\pi i/8}$). Note that $\omega^2 = \varpi$.

If we apply F_4 to $|a_2a_1\rangle$ (the two higher order digits of a), the result is

$$F_4 |a_2a_1\rangle = \Big(|0\rangle + \varpi^{2[a_2a_1]} |1\rangle\Big) \otimes \Big(|0\rangle + \varpi^{[a_2a_1]} |1\rangle\Big) \tag{210}$$

$$= \Big(|0\rangle + (\omega^2)^{2[a_2a_1]} |1\rangle\Big) \otimes \Big(|0\rangle + (\omega^2)^{[a_2a_1]} |1\rangle\Big) \tag{211}$$

$$= \Big(|0\rangle + \omega^{2[a_2a_10]} |1\rangle\Big) \otimes \Big(|0\rangle + \omega^{[a_2a_10]} |1\rangle\Big). \tag{212}$$

In the exponent, I've surrounded the binary representations by square brackets so that they can be clearly read; the two-digit number in the square brackets is either 0, 1, 2, or 3. Eq. (210) is due to Lemma 19.1. Eq. (211) is due to $\varpi = \omega^2$. And Eq. (212) is due[29] to $2[a_2a_1] = [a_2a_10]$.

Now let's express $|\psi\rangle$ in terms of $[a_2a_1a_0]$. This is

$$|\psi\rangle = \left(|0\rangle + \omega^{2[a_2a_1a_0]}\,|1\rangle\right) \otimes \left(|0\rangle + \omega^{[a_2a_1a_0]}\,|1\rangle\right) \tag{213}$$

$$= \left(|0\rangle + \omega^{2[a_2a_10]}\omega^{2a_0}\,|1\rangle\right) \otimes \left(|0\rangle + \omega^{[a_2a_10]}\omega^{a_0}\,|1\rangle\right). \tag{214}$$

Comparing Eq. (212) with Eq. (214), we can see precisely where they differ: the factors ω^{2a_0} and ω^{a_0} (highlighted in red). We'll refer to these as *phase corrections*.

Based on the above observations, let's try to compute $F_8\,|a_2a_1a_0\rangle$ in terms of $F_4\,|a_2a_1\rangle$ and $H\,|a_0\rangle$, combined with additional gates that perform phase corrections. To perform the phase corrections, we introduce the following new 2-qubit gate.

Definition 19.3 (controlled-phase gate)**.** For any $r \in \mathbb{Z}$, the 2-qubit *controlled-phase gate (with phase $e^{2\pi i/r}$)* is defined as the unitary operation that, for all $a, b \in \{0, 1\}$, maps $|a\rangle\,|b\rangle$ to $(e^{2\pi i/r})^{ab}\,|a\rangle\,|b\rangle$. The unitary matrix for the gate is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{2\pi i/r} \end{bmatrix} \tag{215}$$

and our circuit notation for this gate is shown in figure 142.

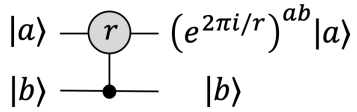

Figure 142: Our notation for the controlled-phase gate, with phase $e^{2\pi i/r}$.

In the special case where $m = 2$, this is a controlled-Z gate. We will employ these gates with r set to higher powers of two ($r = 4, 8, 16, \ldots$).

Now we'll analyze the following circuit and show that it computes F_8.

[29] This is a simple maneuver. It's the binary equivalent of what we do in base ten when we multiply an integer by ten: we add a zero digit and shift all the other digits left. 10 times 23 is 230.

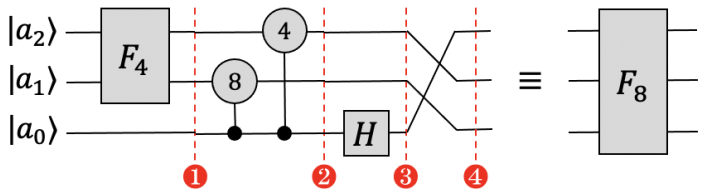

Figure 143: Caption.

We will trace through the stages of this circuit.

State at stage ❶

From Eq. (212), this is $\left(|0\rangle + \omega^{2[a_2a_10]}\,|1\rangle\right) \otimes \left(|0\rangle + \omega^{[a_2a_10]}\,|1\rangle\right) \otimes |a_0\rangle$.

State at stage ❷

The two controlled-phase gates change the state to

$$\left(|0\rangle + \omega^{2[a_2a_10]}\omega^{2a_0}\,|1\rangle\right) \otimes \left(|0\rangle + \omega^{[a_2a_10]}\omega^{a_0}\,|1\rangle\right) \otimes |a_0\rangle \tag{216}$$

$$= \left(|0\rangle + \omega^{2[a_2a_1a_0]}\,|1\rangle\right) \otimes \left(|0\rangle + \omega^{[a_2a_1a_0]}\,|1\rangle\right) \otimes |a_0\rangle \tag{217}$$

$$= \left(|0\rangle + \omega^{2a}\,|1\rangle\right) \otimes \left(|0\rangle + \omega^{a}\,|1\rangle\right) \otimes |a_0\rangle\,. \tag{218}$$

State at stage ❸

Applying a Hadamard gate to the third qubit changes the state to

$$\left(|0\rangle + \omega^{2a}\,|1\rangle\right) \otimes \left(|0\rangle + \omega^{a}\,|1\rangle\right) \otimes \left(|0\rangle + (-1)^{a_0}\,|1\rangle\right) \tag{219}$$

$$= \left(|0\rangle + \omega^{2a}\,|1\rangle\right) \otimes \left(|0\rangle + \omega^{a}\,|1\rangle\right) \otimes \left(|0\rangle + \omega^{4a}\,|1\rangle\right). \tag{220}$$

Notice that this is the state in Eq. (208), except the qubits are in the wrong order.

State at stage ❹

Moving the third qubit to the left and shifting the other two qubits right yields

$$\left(|0\rangle + \omega^{4a}\,|1\rangle\right) \otimes \left(|0\rangle + \omega^{2a}\,|1\rangle\right) \otimes \left(|0\rangle + \omega^{a}\,|1\rangle\right) = F_8\,|a\rangle\,. \tag{221}$$

What we have shown is a special case of the following more general recurrence for F_{2^n}.

Theorem 19.1. *For all $n \geq 1$, the Fourier transform F_{2^n} can be expressed as*

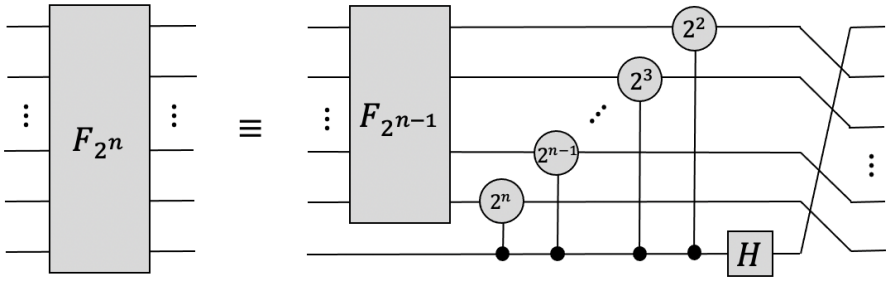

Figure 144: Quantum circuit for F_{2^n} recursively constructed in terms of $F_{2^{n-1}}$.

This is straightforward to verify, along the lines of the analysis for the $n = 3$ case.

19.3.2 Unravelling the recurrence

By repeatedly applying Theorem 19.1, we can construct a quantum circuit for the Fourier transform for any $n \geq 1$. The recurrence bottoms out at $n = 1$, where $F_2 = H$. Here is the circuit for the $n = 4$ case.

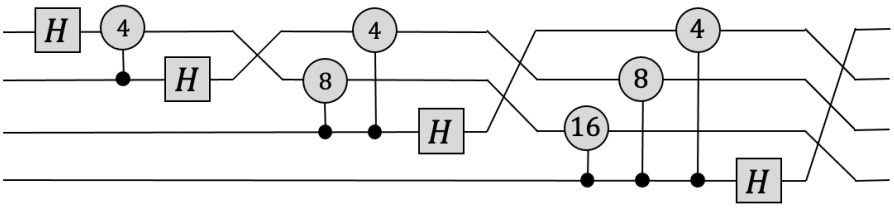

Figure 145: Quantum circuit for F_{2^4}.

Notice that there are places where the qubits are rearranged. Instead of doing these rearrangements, we can move the gates around. Then there's just one net rearrangement at the very end, which turns out to be a reversal of the order of the qubits.

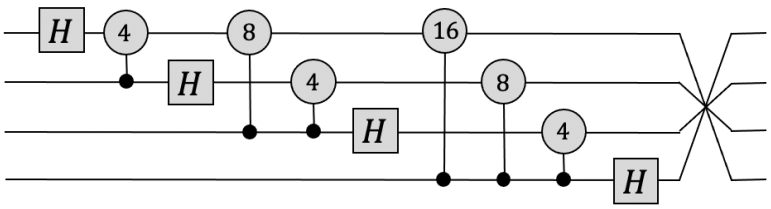

Figure 146: Quantum circuit for F_{2^4}, with rearrangements deferred until the end.

How do we perform this reversal of the order of the qubits? We can define a 2-qubit SWAP gate.

Definition 19.4 (SWAP gate)**.** The 2-qubit *SWAP gate* is defined as the unitary operation that, for all $a, b \in \{0, 1\}$, maps $|a\rangle\,|b\rangle$ to $|b\rangle\,|a\rangle$. The unitary matrix for the gate is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{222}$$

and circuit notation for this gate is shown in figure 147.

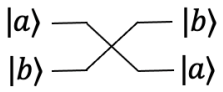

Figure 147: Notation for SWAP gate.

The reversal consists of around $n/2$ SWAP gates.

Intuitively, the notation for SWAP gate suggests a physically movement of the qubits. In principle, one could do that, but it would be nice not to be adding a brand new elementary operation into our repertoire of 2-qubit gates (if we can avoid it). It turns out this this SWAP gate can be computed by three CNOT gates.

Exercise 19.5. Show that the 2-qubit SWAP gate can be implemented by a sequence of three CNOT gates (appropriately oriented).

Finally, let's count the number of gates in our circuit construction (which is the natural generalization of the construction in figure 146 to F_{2^n}). There are n Hadamard gates, one for each qubit. The number of controlled-phase gates is

$$1 + 2 + 3 + \cdots + (n-1) = \frac{n(n-1)}{2} = O(n^2). \tag{223}$$

And there are most $\frac{n}{2}$ SWAP gates—which translates into $\frac{3n}{2}$ CNOT gates. That's a total of $O(n^2)$ gates for computing the Fourier transform F_{2^n}.

Exercise 19.6 (straightforward)**.** Give a quantum circuit that computes the inverse Fourier transform $F_{2^n}^*$ with $O(n^2)$ gates. (There are two different approaches that lead to slightly different circuits.)

19.4 Computing the Fourier transform for arbitrary modulus

What about the case where the modulus m is not a power of 2? Here we provide some references (in the footnotes) to circuit constructions of the Fourier transform for the case where the modulus is not a power of 2:

- There are some fairly simple constructions for the special case where the modulus is a product of "small" prime factors.[30]
- The case of an *arbitrary* modulus is more challenging and has been addressed.[31]

Neither reference explicitly states bounds on the number of gates, but it is clear from the constructions that the gate counts are polynomial in n, where n is the number of bits of the modulus.

[30] R. Cleve (1994). "A note on computing Fourier transforms by quantum programs." unpublished.

[31] M. Mosca and Ch. Zalka (2004). "Exact quantum Fourier transforms and discrete logarithm algorithms." *International Journal of Quantum Information*, **2**(1): 91–100. arXiv:0301093

20 Definition of the discrete log problem

The quantum "algorithms" that we've seen up until now are not algorithms in usual sense. They are in the black-box model, where one is given an unknown function and the goal is to extract information about the function with as few queries to the function as possible. Often, the unknown function is promised to have a special kind of structure (such as the function arising in Simon's problem).

In the next section I will describe a remarkable algorithm for solving the *discrete log problem (DLP)*. This is not a black box problem! It is a conventional computational problem where the input is given as a binary string and the output is also a binary string. No classical polynomial algorithm is known for this problem. In fact, the presumed hardness of this problem has been the basis of cryptosystems (such as the Diffie-Hellman key exchange protocol). In 1994, Peter Shor discovered a polynomial quantum algorithm for this problem, and it's based on the underlying methodologies used in Simon's algorithm—which may sound surprising, since Simon's problem is a black-box problem.

In this section, I define the discrete log problem. In the next section I will describe Shor's polynomial quantum algorithm for this problem. In order to define the discrete log problem, we first need to be familiar with two sets, called $\mathbb{Z}_m$ and $\mathbb{Z}_m^*$.

20.1 Definitions of $\mathbb{Z}_m$ and $\mathbb{Z}_m^*$

Definition 20.1 (of the ring $\mathbb{Z}_m$)**.** For any positive integer $m \geq 2$, define $\mathbb{Z}_m$ as the set $\{0, 1, \ldots, m-1\}$, equipped with addition and multiplication modulo m.

A more familiar example of a ring is the set of all integers $\mathbb{Z}$ equipped with "ordinary" addition and multiplication. Note that there are only two elements of $\mathbb{Z}$ that have multiplicative inverses (namely $+1$ and -1). What are the elements of $\mathbb{Z}_m$ that have multiplicative inverses? It depends on m. Let's look at the case where $m = 9$. The elements of $\mathbb{Z}_9$ that have multiplicative inverses are 1, 2, 4, 5, 7, 8 (0, 3, and 6 do not have multiplicative inverses).

Definition 20.2 (of the group $\mathbb{Z}_m^*$)**.** For any positive integer $m \geq 2$, the set $\mathbb{Z}_m^*$ consists of all elements of $\mathbb{Z}_m$ that have multiplicative inverses. $\mathbb{Z}_m^*$ is a group.

For the purposes of DLP, we need only consider $\mathbb{Z}_p^*$ for prime p. In that case, it's known that every non-zero element of $\mathbb{Z}_p^*$ has a multiplicative inverse. Therefore we can use this simple definition of the group $\mathbb{Z}_p^*$ for case where p is prime.

Definition 20.3 (of the group $\mathbb{Z}_p^*$)**.** For any prime p, define $\mathbb{Z}_p^*$ as the group with elements $\{1, \ldots, p-1\}$, where the operation is multiplication modulo p.

20.2 Generators of $\mathbb{Z}_p^*$ and the exponential/log functions

An element g of the group $\mathbb{Z}_p^*$ is called a *generator* of the group if the set of all powers g is the group. Whenever a group has such an element, it is called *cyclic*.

Let's consider the case where $p = 7$ as an example. $\mathbb{Z}_7^* = \{1, 2, 3, 4, 5, 6\}$. Obviously 1 is not a generator, since all powers of 1 are just 1. What about 2? 2 is not a generator either, because of we list the powers of 2, which are 2^0, 2^1, 2^2, and so on, we get the sequence $1, 2, 4, 1, 2, 4, \ldots$ so we only get the set $\{1, 2, 4\}$, which is a proper subgroup of $\mathbb{Z}_7^*$. What about 3? 3 *is* a generator, since the powers of 3 are $1, 3, 2, 6, 4, 5, 1, 3, 2, \ldots$, which is the entire set $\mathbb{Z}_7^*$. It turns out that $\mathbb{Z}_p^*$ is cyclic for any prime p.

Theorem 20.1. *For any prime p, there exists $g \in \mathbb{Z}_p^*$ such that*

$$\mathbb{Z}_p^* = \left\{ g^k : k \in \{0, 1, \ldots, p-2\} \right\}. \tag{224}$$

Notice that the exponents run from 0 to $p-2$. This makes sense because the size of $\mathbb{Z}_p^*$ is $p-1$ (it's not p because $0 \notin \mathbb{Z}_p^*$). So the set of exponents of a generator is $\mathbb{Z}_{p-1}$ (it's not $\mathbb{Z}_p$). And, if the elements of $\mathbb{Z}_p^*$ are expressed as powers of a generator g, then $g^x g^y = g^{x+y \bmod p-1}$ holds for any $x, y \in \mathbb{Z}_{p-1}$. It follows that multiplication in $\mathbb{Z}_p^*$ corresponds to addition mod $p-1$ in the exponents of g. Formally, there is an isomorphism between the additive group $\mathbb{Z}_{p-1}$ and the multiplicative group $\mathbb{Z}_p^*$.

Definition 20.4 (discrete exp function)**.** Relative to a prime modulus p and a generator g of $\mathbb{Z}_p^*$, define the *discrete exponential function* $\exp_g : \mathbb{Z}_{p-1} \to \mathbb{Z}_p^*$ as, for all $r \in \mathbb{Z}_{p-1}$,

$$\exp_g(r) = g^r. \tag{225}$$

Definition 20.5 (discrete log function)**.** Relative to a prime modulus p and a generator g of $\mathbb{Z}_p^*$, define the *discrete log function* $\log_g : \mathbb{Z}_p^* \to \mathbb{Z}_{p-1}$ as the inverse of the discrete exponential function $\exp_g$. The input to $\log_g$ is some $s \in \mathbb{Z}_p^*$, and the output is the value of $r \in \mathbb{Z}_{p-1}$ such that $g^r = s$.

20.3 Discrete exponential problem

For the *discrete exp problem*, the input is (p, g, r), where

- p is an n-bit prime.
- g is a generator of $\mathbb{Z}_p^*$.
- $r \in \mathbb{Z}_{p-1}$.

And the output is: $\exp_g(r)$, which is g^r.

How hard is it to compute this function? Of course, g^r is equal to g multiplied by itself r times. So $\exp_g(r)$ can obviously be computed by r multiplications (the precise number of multiplications is actually $r-1$). But r is an n-bit number, so r can be roughly as large as 2^n. That's an exponentially large number of multiplication operations! Using this approach, the circuit-size would be exponential in n. But there's a simple trick for doing this more efficiently, called the *repeated squaring trick*.

20.3.1 Repeated squaring trick

The idea is the following. You multiply g by itself to get g^2. Then you multiply g^2 by itself to get g^4. And then you multiply g^4 by itself get g^8, and so on. This way, you can compute g^{2^n} at the cost of only n multiplications. That's how the repeated squaring trick works when the exponent r is a power of 2.

What if r is not a power of 2? The above idea can be adjusted to compute *any* n-bit exponent r with fewer than $2n$ multiplications. The idea is based on the fact that g^r can be written as

$$g^{r_n \cdots r_3 r_2 r_1} = ((\cdots(g^{r_n})^2 \cdots g^{r_3})^2 g^{r_2})^2 g^{r_1}, \tag{226}$$

where $r_n ... r_3 r_2 r_1$ is the binary representation of an n-bit exponent r.

Note that $O(n)$ multiplications, at cost $O(n \log(n))$ gates each, leads to a classical gate cost of order $O(n^2 \log(n))$ for computing the discrete exponential function.

20.4 Discrete log problem

For the *discrete log problem (DLP)*, the input is (p, g, s), where

- p is an n-bit prime.
- g is a generator of $\mathbb{Z}_p^*$.
- $s \in \mathbb{Z}_p^*$.

And the output is: $\log_g(s)$, which is the $r \in \mathbb{Z}_{p-1}$ for which $g^r = s$.

How hard is it to compute this function? Currently, there is no known classical algorithm that solves DLP with polynomially many gates (with respect to the input size n).

21 Shor's algorithm for the discrete log problem

The overall idea behind Shor's algorithm[32] is to convert the discrete log problem (DLP) into a generalization of Simon's problem, and then to solve that generalization of Simon's problem. Since DLP is not a black-box problem, how can such a conversion work? It works by creating a function with a property similar to Simon's *that can be efficiently implemented* so as to simulate black-box queries to the function. This is Shor's function.

21.1 Shor's function with a property similar to Simon's

Recall that an instance of the discrete log problem consists of three n-bit numbers: p (an n-bit prime), g (a generator of $\mathbb{Z}_p^*$), and s (an element of $\mathbb{Z}_p^*$).

Shor's function $f : \mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1} \to \mathbb{Z}_p^*$ is defined as

$$f(a_1, a_2) = g^{a_1} s^{-a_2}, \tag{227}$$

for all $a_1, a_2 \in \mathbb{Z}_{p-1}$. What's interesting about this function is where the collisions are. When is $f(a_1, a_2) = f(b_1, b_2)$?

Theorem 21.1. *For an instance (p, g, s) of DLP, the function $f : \mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1} \to \mathbb{Z}_p^*$ defined by Eq. (227) has the property that*

$$f(a_1, a_2) = f(b_1, b_2) \quad \textit{if and only if } (a_1, a_2) - (b_1, b_2) \textit{ is a multiple of } (r, 1), \tag{228}$$

where $r = \log_g(s)$.

The significance of this is that it resembles the Simon property. It's the analogue of the Simon property when we switch from mod 2 arithmetic to mod $p-1$ arithmetic. To see this, recall that the Simon property can be stated as: $f(a) = f(b)$ if and only if $a \oplus b$ is either a string of n zeroes or the string r. Notice that: $a \oplus b$ is the same as $a - b$ in mod 2 arithmetic. So we can write the Simon property as:

> **Simon property for** $f : (\mathbb{Z}_2)^n \to (\mathbb{Z}_2)^n$
> $f(a_1, \ldots, a_n) = f(b_1, \ldots, b_n)$ if and only if $(a_1, \ldots, a_n) - (b_1, \ldots, b_n)$ is a multiple of $(r_1, \ldots, r_n)$ in mod 2 arithmetic.

[32] P. W. Shor (1994). "Algorithms for quantum computation: discrete logarithms and factoring." *Proc. 35th Annual Symp. on Foundations of Computer Science*, pp. 124–134. arXiv:9508027

And here again is the property that Shor's function has:

> **Property of Shor's function for** $f : \mathbb{Z}_{p-1} \times \mathbb{Z}_{p-1} \to \mathbb{Z}_p^*$
> $f(a_1, a_2) = f(b_1, b_2)$ if and only if $(a_1, a_2) - (b_1, b_2)$ is a multiple of $(r, 1)$ in mod $p - 1$ arithmetic.

Proof of Theorem 21.1. The proof is elementary, but we'll go through it carefully. Although we do not know what r is, we do know that an r exists such that $s = g^r$. This means that $s^{a_2} = g^{ra_2}$. Therefore, $f(a_1, a_2) = g^{a_1} g^{-ra_2} = g^{a_1 - ra_2}$. It's nice to write f this way, because then

$$f(a_1, a_2) = f(b_1, b_2) \tag{229}$$

$$\text{if and only if} \quad a_1 - ra_2 = b_1 - rb_2 \tag{230}$$

$$\text{if and only if} \quad (a_1, a_2) \cdot (1, -r) = (b_1, b_2) \cdot (1, -r) \tag{231}$$

$$\text{if and only if} \quad \Big((a_1, a_2) - (b_1, b_2)\Big) \cdot (1, -r) = 0. \tag{232}$$

In equation (232), the dot-product being zero is like an orthogonality relation between the vector $(a_1, a_2) - (b_1, b_2)$ and the vector $(1, -r)$. Here's a schematic sketch of the vector $(1, -r)$.

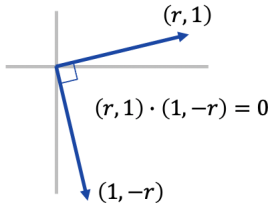

Figure 148: Schematic illustration of $(r, 1)$ orthonormal to $(1, -r)$.

Notice that the vector $(r, 1)$ is orthogonal to the vector $(1, -r)$ in that their dot-product is zero. Now, in a two-dimensional space, the vectors that are orthogonal to a particular vector are all multiples of *one* of the orthogonal vectors. So, we might expect $(a_1, a_2) - (b_1, b_2)$ to be a multiple of $(r, 1)$. This intuition (valid for vector spaces with inner products) is confirmed in our context by a simple calculation:

$$(v_1, v_2) \cdot (1, -r) = 0 \tag{233}$$

$$\text{if and only if} \quad v_1 = rv_2 \tag{234}$$

$$\text{if and only if} \quad v_2 = k \ \text{ and } \ v_1 = rk, \ \text{ for some } k \in \mathbb{Z}_{p-1} \tag{235}$$

$$\text{if and only if} \quad (v_1, v_2) = k(r, 1), \ \text{ for some } k \in \mathbb{Z}_{p-1}. \tag{236}$$

Therefore, $f(a_1, a_2) = f(b_1, b_2)$ if and only if $(a_1, a_2) - (b_1, b_2)$ is a multiple of $(r, 1)$. ■

21.2 The Simon mod m problem

We've established that Shor's function satisfies a variant of Simon's property where the domain of the function is changed from $\{0,1\}^n = (\mathbb{Z}_2)^n$ to $(\mathbb{Z}_{p-1})^2$.

Now, what we're going to do is digress from DLP to investigate the generalization of Simon's problem, where the modulus is changed from 2 to m. We'll first find an efficient quantum black-box algorithm for the Simon mod m problem, where we are given a function

$$f : (\mathbb{Z}_m)^d \to T, \tag{237}$$

where T can be any set—but for convenience we'll assume that $T = \{0,1\}^k$ for some k (any T can be enlarged so that it's size is a power of 2).

Definition 21.1 (of an m-to-1 function)**.** A function is *m-to-1* if, for every value attained by the function, there are exactly m preimages. In other words, all colliding sets are of size m.

Recall that, in the Simon's original problem, we had colliding pairs. Now, here is a mod m analogue of the original Simon property.

Definition 21.2 (Simon mod m property)**.** An m-to-1 function $f : (\mathbb{Z}_m)^d \to T$ satisfies the *Simon mod m property*, if there exists a non-zero $r \in (\mathbb{Z}_m)^d$ such that: for all $a, b \in (\mathbb{Z}_m)^d$, it holds that $f(a) = f(b)$ if and only if $a - b$ is a multiple of r.

Notice that the Simon mod m property is equivalent to the property that every colliding set of f is of the form $\{a, a+r, a+2r, \ldots, a+(m-1)r\} = \{a+kr : k \in \mathbb{Z}_m\}$, for some $a \in (\mathbb{Z}_m)^d$. Figure 149 is a schematic diagram of the two-dimensional case.

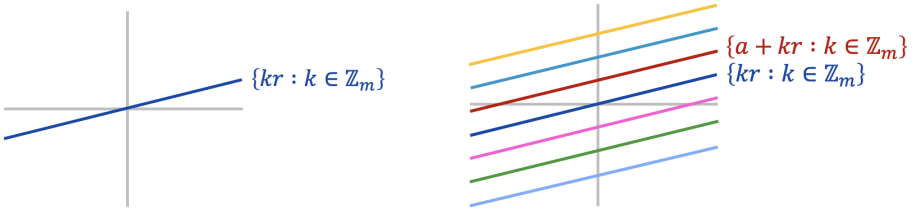

Figure 149: Each colliding set is one of the colored lines (an offset of the line $\{kr : k \in \mathbb{Z}_m\}$).

Think of $\{kr : k \in \mathbb{Z}_m\}$ as a line in $(\mathbb{Z}_m)^d$ passing through the origin. And think of $\{a+kr : k \in \mathbb{Z}_m\}$ as an *offset* of the line by $a \in (\mathbb{Z}_m)^d$. Each colliding set is an offset.

Recall that, for the original Simon property, the colliding pairs were of the form $\{a, a \oplus r\}$, which are offsets of $\{0, r\}$. In that case, the colored lines in figure 149 correspond to the colliding pairs.

Definition 21.3. For a function $f : (\mathbb{Z}_m)^d \to T$, define an f-query as unitary operation U_f such that, for every $(a_1, a_2, \ldots, a_n) \in (\mathbb{Z}_m)^d$ and $b \in T$,

$$U_f\Big(|a_1\rangle\,|a_2\rangle \ldots |a_n\rangle\,|b\rangle\Big) = |a_1\rangle\,|a_2\rangle \ldots |a_n\rangle\,|b \oplus f(a_1, a_2, \ldots, a_n)\rangle\,, \tag{238}$$

where the $\oplus$ operation denotes bit-wise XOR (recall our assumption that $T = \{0,1\}^k$).

We use the following notation for f-queries.

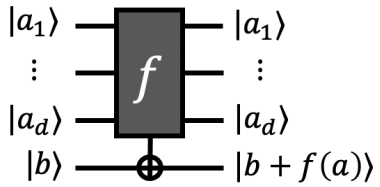

Figure 150: Notation for a quantum f-query gate for $f : (\mathbb{Z}_m)^d \to T$.

Notice that the lines are drawn thicker than those in our previous quantum circuits where the lines carried qubits. This is to help convey the fact that the data flowing through these lines is generally not qubits, but states of dimension higher than 2. The first d lines are carrying an m-dimensional quantum states, and the last line is carrying a $|T|$-dimensional state.

Definition 21.4 (Simon mod m problem)**.** For Simon's problem mod m, you're given a black-box that computes an f-query for an unknown function f that is promised to have the Simon mod m property, and your goal is to determine the parameter r, based on queries to f.

How do we solve the Simon mod m problem? Let's start by recalling the algorithm for the original Simon's problem. It was based on repeated runs of this circuit that makes a single f-query.

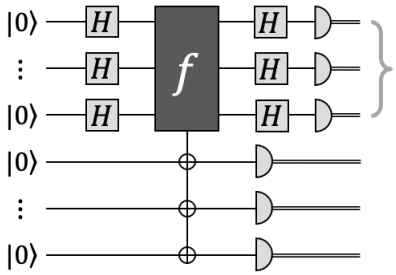

Figure 151: Circuit used for Simon's problem.

Each run of the circuit produces a uniformly random element of the orthogonal complement of r. After a few runs, we obtain enough such b's to be able to deduce r by solving a system of linear equations in mod 2 arithmetic.

The proposed algorithm for the Simon mod m problem will be based on a quantum circuit similar to the one in figure 151, but where the horizontal lines represent m-dimensional registers (rather than qubits) and the single-register gates are the Fourier transforms defined in section 19.1 and their inverses (rather than Hadamard transforms).

21.3 Query algorithm for Simon mod m

The algorithm is based on the circuit in figure 152.

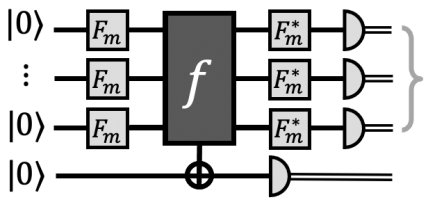

Figure 152: Proposed circuit for the Simon mod m problem.

Note that, in the special case where $m = 2$, the circuit in figure 152 is almost the same as the circuit in figure 151. What's the output of this circuit? We'll analyze this for the case where $d = 2$, which the circuit in figure 153.

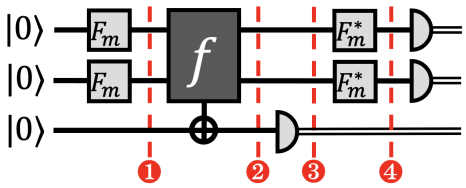

Figure 153: Circuit for Simon mod m problem in the case where $d = 2$.

Let's trace through the evolution of the state of the registers.

State at stage ❶

Since $F_m |0\rangle = \frac{1}{\sqrt{m}}(|0\rangle + |1\rangle + \cdots + |m-1\rangle)$, this state is

$$\frac{1}{\sqrt{m^2}} \sum_{a \in \mathbb{Z}_m \times \mathbb{Z}_m} |a_1, a_2\rangle |0\rangle . \tag{239}$$

State at stage ❷

Since the query maps each basis state $|a_1, a_2\rangle |0\rangle$ to $|a_1, a_2\rangle |f(a_1, a_2)\rangle$, when the input is superposition, the output of the query is

$$\frac{1}{\sqrt{m^2}} \sum_{a \in \mathbb{Z}_m \times \mathbb{Z}_m} |a_1, a_2\rangle |f(a_1, a_2)\rangle . \tag{240}$$

State of the first two registers at stage ❸

Measuring the state of the third register causes its state to collapse to some value of f and the state of the first two registers to collapse to a uniform superposition of all the pre-images of that value of f. This preimage is one of the collapsing sets of f (one of the colored lines in figure 149). Therefore, the state after this measurement is

$$\frac{1}{\sqrt{m}} \sum_{k \in \mathbb{Z}_m} |(a_1, a_2) + k(r_1, r_2)\rangle , \tag{241}$$

for some $(a_1, a_2) \in \mathbb{Z}_m \times \mathbb{Z}_m$.

This state is identical to the state that is the outcome of the following process:

1. Randomly choose one of the colliding sets (the colored lines in figure 149).
2. Take the uniform superposition of the elements of that colliding set.

State of the first two registers at stage ❹

$$F_m^* \otimes F_m^* \left(\tfrac{1}{\sqrt{m}} \sum_{k \in \mathbb{Z}_m} |(a_1, a_2) + k(r_1, r_2)\rangle \right) \tag{242}$$

$$= \tfrac{1}{\sqrt{m}} \sum_{k \in \mathbb{Z}_m} \left(F_m^* \otimes F_m^* |(a_1, a_2) + k(r_1, r_2)\rangle \right) \tag{243}$$

$$= \tfrac{1}{\sqrt{m}} \sum_{k \in \mathbb{Z}_m} \left(\tfrac{1}{\sqrt{m^2}} \sum_{b \in \mathbb{Z}_m \times \mathbb{Z}_m} \omega^{-(a+kr)\cdot b} |b_1, b_2\rangle \right) \tag{244}$$

$$= \tfrac{1}{\sqrt{m}} \sum_{k \in \mathbb{Z}_m} \left(\tfrac{1}{m} \sum_{b \in \mathbb{Z}_m \times \mathbb{Z}_m} \omega^{-a\cdot b} \omega^{-k(r\cdot b)} |b_1, b_2\rangle \right) \tag{245}$$

$$= \tfrac{1}{\sqrt{m}} \sum_{b \in \mathbb{Z}_m \times \mathbb{Z}_m} \omega^{-a\cdot b} \left(\tfrac{1}{m} \sum_{k \in \mathbb{Z}_m} \omega^{-k(r\cdot b)} \right) |b_1, b_2\rangle \,. \tag{246}$$

Notice that Eq. (246) contains an expression for the amplitude of $|b_1, b_2\rangle$ for any $(b_1, b_2) \in \mathbb{Z}_m \times \mathbb{Z}_m$. Since $|\omega^{-a\cdot b}| = 1$, what's important in Eq. (246) is the expression in parentheses, which is

$$\tfrac{1}{m} \sum_{k \in \mathbb{Z}_m} \omega^{-k(r\cdot b)} = \begin{cases} 1 & \text{if } (r_1, r_2) \cdot (b_1, b_2) = 0 \\ 0 & \text{if } (r_1, r_2) \cdot (b_1, b_2) \neq 0. \end{cases} \tag{247}$$

This implies that, for any $(b_1, b_2) \in \mathbb{Z}_m \times \mathbb{Z}_m$,

$$\Pr[(b_1, b_2)] = \begin{cases} \frac{1}{m} & \text{if } (r_1, r_2) \cdot (b_1, b_2) = 0 \\ 0 & \text{if } (r_1, r_2) \cdot (b_1, b_2) \neq 0. \end{cases} \tag{248}$$

Therefore the output of the circuit in figure 153 is a uniformly random sample from the set $\{(b_1, b_2) \in \mathbb{Z}_m \times \mathbb{Z}_m : b_1 r_1 + b_2 r_2 = 0\}$, which we loosely call the *orthogonal complement* of (r_1, r_2).

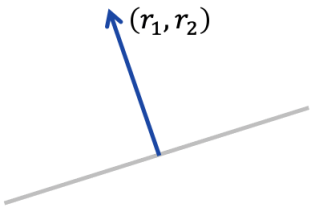

Figure 154: Schematic illustration of the orthogonal complement of (r_1, r_2).

All of the preceeding analysis generalizes in a straightforward way from two dimensions to d dimensions.

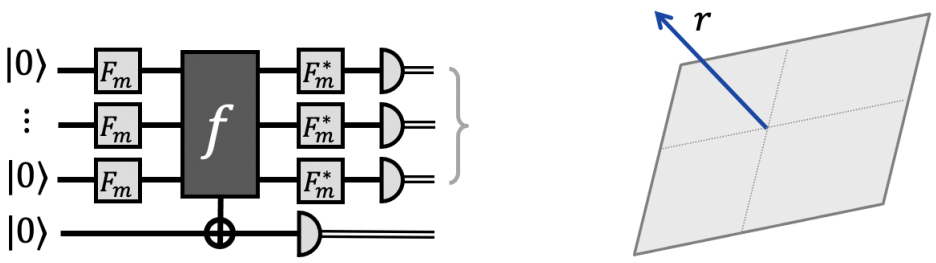

Figure 155: Simon mod m circuit produces a random element of the orthogonal complement of r.

In that case, output from the first d registers is a uniformly random element of the set $\{b \in \mathbb{Z}_m^d : b \cdot r = 0\}$ (the orthogonal complement of r).

As with Simon's algorithm, after repeated runs of the circuit in figure 155, we obtain several b's, from which we can hope to deduce r by solving a system of linear equations in mod m arithmetic (we'll return to this matter of solving the linear equation(s) in the context of the discrete log problem in section 21.5.1).

21.4 Returning to the discrete log problem

Now that we've analyzed the Simon mod m problem, let's get back to the original problem that we're concerned with, which is the discrete log problem. The input is: p, an n-bit prime number; g, a generator of $\mathbb{Z}_p^*$; and s, an element of $\mathbb{Z}_p^*$ (all three inputs are binary strings of length n). And the goal to produce $\log_g(s)$, which is the unique $r \in \mathbb{Z}_{p-1}$ for which $s = g^r$.

The reason why we turned our attention to the Simon mod m problem is that there is a special Shor function that satisfies the Simon mod m property (with m set to $p-1$), and a solution to that instance of Simon mod $p-1$ yields $r = \log_g(s)$.

How can we use our solution to Simon mod m to solve an instance of DLP, which is not in the query framework? The idea is to efficiently *implement* the query gate of the Shor function, as well as the other parts of the query circuit in terms of qubits and elementary gates (1- and 2-qubit gates). The elements of $\mathbb{Z}_{p-1}$ and $\mathbb{Z}_p^*$ can be represented by their n-bit binary representations as n-bit strings. Imagine a $3n$-qubit quantum circuit of the form of figure 156, where each grey box represents a quantum circuit consisting of elementary gates acting on qubits.

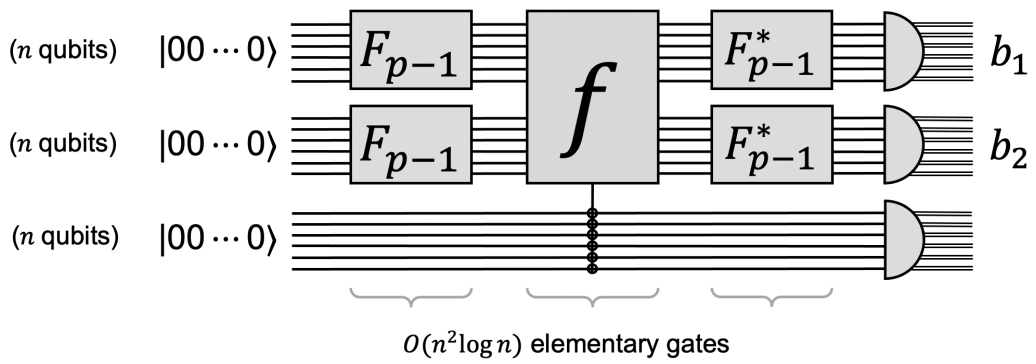

Figure 156: Implementation of the query algorithm in figure 157.

This circuit is an implementation of the circuit in figure 157.

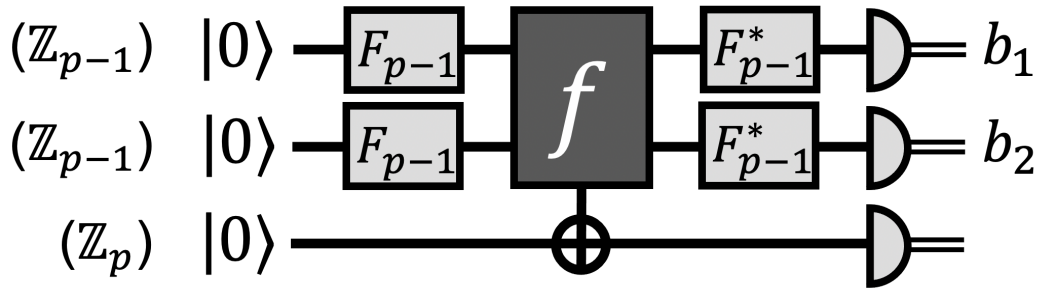

Figure 157: This is what is simulated by the circuit in figure 156.

A property of the Shor function f (defined is Eq. (227)) that's crucial to make this work is that $f(a_1, a_2) = g^{a_1} s^{a_2}$ can be computed *efficiently* by a classical circuit. The exponentiations can be computed efficiently using the repeated squaring trick (explained in section 20.3.1).

That's the overall idea behind Shor's algorithm for the discrete log problem. However, there are a few loose ends that have not been explained.

21.5 Loose ends

One loose end is that concerns technicalities in extracting r from repeated runs of the circuit in figure 153. This is discussed in section 21.5.1.

Another loose end is an important issue that arises in implementing the f-query, which is discussed in sections 21.5.2 and 21.5.3.

Finally, there's the matter of efficiently computing F_m, which is discussed in section 21.5.4.

21.5.1 How to extract r

As was done for the original Simon problem, we can set up a system of linear equations in mod m arithmetic. However, a complication arises when $\mathbb{Z}_m$ is not a field—and in fact, it's *not* a field in our case of interest, where $m = p - 1$, for a prime p.

Recall that, for DLP, the quantum circuit in figure 153 produces a uniformly random (b_1, b_2) such that

$$(b_1, b_2) \cdot (r, 1) \equiv 0 \pmod{p-1}. \tag{249}$$

How do we calculate r from this? From Eq. (249), we can solve for r as

$$r = -b_2/b_1 \bmod p - 1. \tag{250}$$

But, for this to work, b_1 must have an inverse modulo $p - 1$. It might not.

It can be proven that the fraction of elements of $\mathbb{Z}_{p-1}$ that have inverses is not too small. I won't get into further details here about how this is quantified. But, as a result of this, we can simply repeat the process of running the circuit in figure 156 a few times until we obtain a (b_1, b_2) where b_1 has an inverse in $\mathbb{Z}_{p-1}$.

21.5.2 How *not to* compute an f-query

Suppose that there is an efficient classical circuit that computes a function f. How do we simulate an f-query (as in Definition 21.3) in terms of elementary operations on qubits?

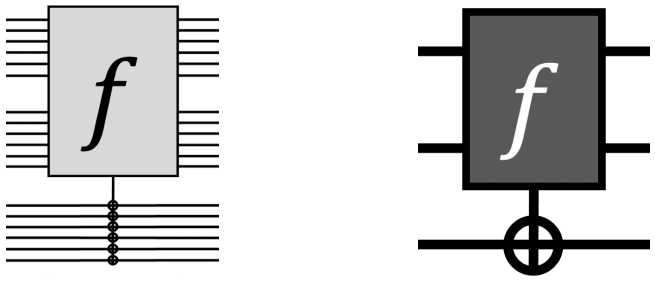

Figure 158: A circuit on qubits (left) so as to simulate an f-query (right).

We saw back in section 14.2 that any classical circuit can be simulated by a quantum circuit with the same efficiency (up to a constant factor). However, that simulation used ancilla qubits and resulted in a quantum quantum circuit that maps each computational basis state of the form $|a\rangle |0^m\rangle |b\rangle$ to $|a\rangle |g(a)\rangle |f(a) \oplus b\rangle$, where $|0^m\rangle$

is a string of several $|0\rangle$ qubits that are used as ancillas, and $g(a)$ consists of some intermediate results of the computation.

To help clarify the point, here again in the quantum circuit from figure 101 (in section 14.2), for computing the majority of three bits.

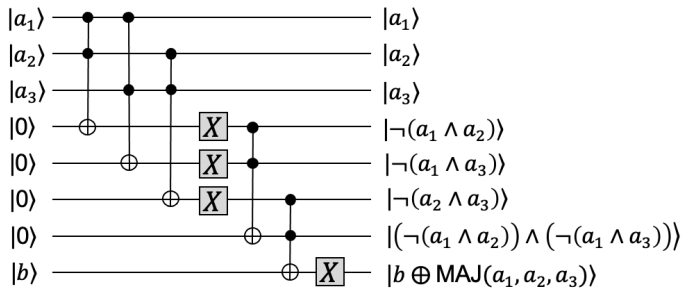

Figure 159: Quantum circuit that simulates classical circuit for the majority function.

The first three qubits contain the input to the function. The last qubit is where the output is XORed. And there are four ancilla qubits, whose states at the end of the computation are shown. We can refer to this as the "garbage", because we might think of disposing of those four qubits. Or ignoring them.

Is this OK for simulating an f-query? No, this leads to a problem if the f-query is applied at a state that's a superposition of computational basis states, such as

$$\sum_{a,b} \alpha_{a,b} |a\rangle |b\rangle |0^m\rangle . \tag{251}$$

The above purported implementation of an f-query maps this state to

$$\sum_{a,b} \alpha_{a,b} |a\rangle |b \oplus f(a)\rangle |g(a)\rangle \neq \left(\sum_{a,b} \alpha_{a,b} |a\rangle |b \oplus f(a)\rangle \right) |g(a)\rangle . \tag{252}$$

The state of the first two registers need not be

$$\sum_{a,b} \alpha_{a,b} |a\rangle |b \oplus f(a)\rangle . \tag{253}$$

(Note that, in Eqns. (251)(252)(253), I've moved the garbage register to the end so that the statement of inequality is simpler to write.)

But this problem has a simple remedy.

21.5.3 How *to* compute an f-query

The first step is to compute f with the ancilla registers. After that, the computation is reversed, so as to restore the ancilla registers back to state $|0\rangle$. But just before the reversal, the answer is XORed to another register, using CNOT gates.

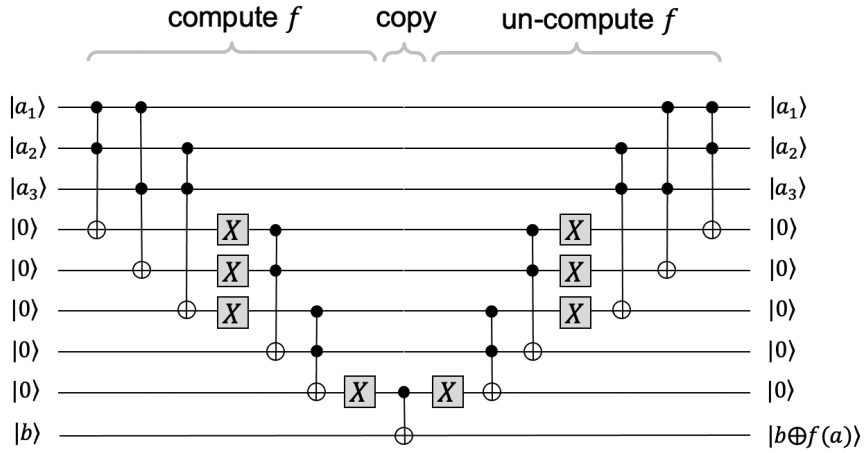

Figure 160: Clean computation of the majority function (the ancilla qubits are reset to $|0\rangle$).

The resulting circuit computes the f-query and restores the ancilla qubits back to their original states. Therefore, this works even if the f-query is applied to a superposition of computational basis states as

$$\sum_{a,b} \alpha_{a,b} |a\rangle |b \oplus f(a)\rangle |0^m\rangle = \left(\sum_{a,b} \alpha_{a,b} |a\rangle |b \oplus f(a)\rangle \right) |0^m\rangle . \tag{254}$$

The garbage register ends end up in a product state with the other registers.

Computing the function this way roughly doubles the number of gates needed.

21.5.4 How to compute the Fourier transform F_{p-1}

Another issue is how to compute the Fourier transform modulo $p-1$. It's an exponentially large matrix, and we need to compute it with a polynomial number of elementary gates acting on n qubits.

In fact, for modulus $p-1$, computing F_{p-1} is tricky (references in section 19.4). Shor's algorithm doesn't actually compute F_{p-1}. Rather, it uses a Fourier transform with modulus a power of 2, which was shown to be easy to compute efficiently in section 19.3. The power of 2 is set so as to be close to $p-1$ (within a factor of 2).

So Shor's algorithm uses the "wrong" Fourier transform. But it's not too wrong. Shor uses careful error analysis to show that, if the modulus is only off by a factor of two then the resulting wrong output state of the circuit is not too wrong. The result of the measurement still succeeds with some constant probability. I won't go into the details of this error analysis here. If you would like to see these details, you can find them in section 6 of the journal version of Shor's paper.[32]

22 Phase estimation algorithm

In this section, we are going to investigate the *phase estimation problem*, which involves finding an eigenvalue of an unknown unitary operation, given as a black-box. This is an abstract problem that turns out to be a powerful algorithmic primitive.

22.1 A simple introductory example

Let U be an n-qubit unitary and $|\psi\rangle$ be an eigenvector of U with eigenvalue either $+1$ or -1. Suppose that we're given a controlled-U gate as a black-box

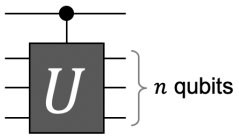

Figure 161: A controlled-U gate as a black box.

and we are also given n qubits that are in state $|\psi\rangle$. We're given no other information about U. Our controlled-U gate is essentially a black-box. Also, we're given no information about what state $|\psi\rangle$ is. All we know is that $|\psi\rangle$ is an eigenvector of U with eigenvalue $\lambda \in \{+1, -1\}$. Our goal is to determine whether λ is $+1$ or -1.

It turns out that we can solve this problem with just one single query to the controlled-U gate. I'd like to you to think about how this can be done. What state should you set the control qubit to? What state should you set the n target qubits to? Now is a good time to stop reading and think about this ...

So how did it go? Did you come up with a quantum circuit for this? If you did not then I'd like to urge you to try again. The solution is a pretty simple circuit. And here is a hint: it's similar to the very first quantum algorithm that we saw, for Deutsch's problem. So please give it another shot ...

Spoiler alert: a solution is on the next page.

Here's a quantum circuit that works.

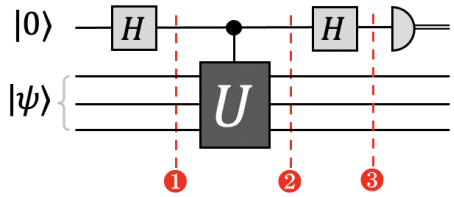

Figure 162: Quantum circuit determining the eigenvalue of U in the special case.

Let's trace through this to see how it works.

State at stage ❶

Applying H to the control qubit results in the state

$$\left(\tfrac{1}{\sqrt{2}}\left|0\right\rangle + \tfrac{1}{\sqrt{2}}\left|1\right\rangle\right)\left|\psi\right\rangle = \tfrac{1}{\sqrt{2}}\left|0\right\rangle\left|\psi\right\rangle + \tfrac{1}{\sqrt{2}}\left|1\right\rangle\left|\psi\right\rangle. \tag{255}$$

State at stage ❷

After the controlled-U gate, the state is

$$\tfrac{1}{\sqrt{2}}\left|0\right\rangle\left|\psi\right\rangle + \tfrac{1}{\sqrt{2}}\left|1\right\rangle U\left|\psi\right\rangle = \tfrac{1}{\sqrt{2}}\left|0\right\rangle\left|\psi\right\rangle + \tfrac{1}{\sqrt{2}}\left|1\right\rangle\lambda\left|\psi\right\rangle \tag{256}$$

$$= \left(\tfrac{1}{\sqrt{2}}\left|0\right\rangle + \tfrac{1}{\sqrt{2}}\lambda\left|1\right\rangle\right)\left|\psi\right\rangle. \tag{257}$$

Notice that, since $\lambda \in \{+1, -1\}$, the control qubit is in either the state $\left|+\right\rangle$ or state $\left|-\right\rangle$ (depending in λ).

State at stage ❸

It follows that, when H is applied again to the control qubit becomes

$$\begin{cases} \left|0\right\rangle & \text{if } \lambda = +1 \\ \left|1\right\rangle & \text{if } \lambda = -1. \end{cases} \tag{258}$$

We have just solved a special case of the *phase estimation problem*.

I will show you how the general case is defined—and then we'll see that it turns out not to be not that much harder than this case to solve. In order to define the phase estimation problem in full generality, we first need to extend our notion of a controlled-U gate.

22.2 Multiplicity controlled-U gates

Let's begin with a review of what a standard controlled-U gate is.

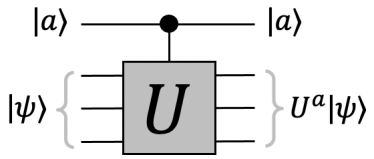

Figure 163: Controlled-U gate.

The unitary matrix is the block matrix

$$\begin{bmatrix} I & 0 \\ 0 & U \end{bmatrix}. \tag{259}$$

When the control qubit is in computational basis state $|a\rangle$ and the target qubit is in state $|\psi\rangle$, we can write the output state of the target qubit succinctly as $U^a |\psi\rangle$ (where $U^0 = I$ means "do nothing", and $U^1 = U$ means "apply U"). So, in the computational basis, the control qubit is a number which indicates how many times U should be applied to the target: 0 times or 1 time.

We can define a more general type of controlled-U gate where there are ℓ control qubits, and the number of times that U is applied is an ℓ-bit integer.

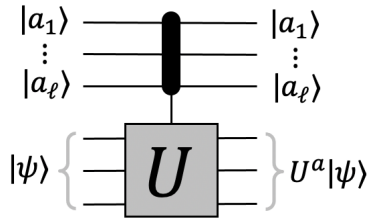

Figure 164: Multiplicity-controlled-U gate.

In this case, U can be applied zero times, U can be applied once, twice, three times, all the way up to $2^\ell - 1$ times. For example, if $\ell = 5$ and the control qubits are in state $|11010\rangle$ then U gets applied 26 times. Why 26? because 11010 is the number 26 in binary.

The unitary matrix of this kind of controlled-U gate is the block matrix

$$\begin{bmatrix} I & 0 & 0 & \cdots & 0 \\ 0 & U & 0 & \cdots & 0 \\ 0 & 0 & U^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & U^{2^\ell - 1} \end{bmatrix}. \tag{260}$$

Let's call this a *multiplicity-controlled-U gate*. In the computational basis, the state of the control qubits specifies how many times U should be applied.

As an aside, I want to distinguish this kind of gate from a different generalization of a controlled-U that we saw previously in the Toffoli gate.

22.2.1 Aside: multiplicity-control gates vs. AND-control gates

For the Toffoli gate, there are two control qubits and the unitary U is the Pauli X gate (a.k.a. NOT gate). For the Toffoli gate, the NOT is applied to the target qubit if both control qubits are $|1\rangle$, and nothing happens for the other computational basis states $|00\rangle$, $|01\rangle$, $|10\rangle$.

Our notation distinguishes between these controlled gates and our multiplicity-controlled gate.

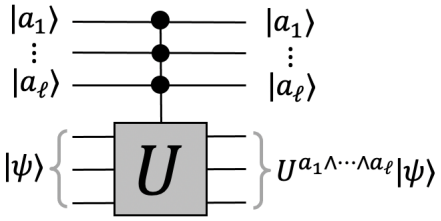

Figure 165: AND-controlled-U gate.

When we have a separate *dot* at each control qubit, that means that the U gets applied once if *all* control qubits are in state $|1\rangle$ and, for other computational basis states, nothing happens. So the unitary matrix is the block matrix

$$\begin{bmatrix} I & 0 & \cdots & 0 & 0 \\ 0 & I & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I & 0 \\ 0 & 0 & \cdots & 0 & U \end{bmatrix}. \tag{261}$$

Our multiplicity-controlled-U gates are denoted differently, with *one single dot*, that's stretched out so as to cover all of the control qubits (figure 164).

22.3 Definition of the phase estimation problem

Now, we can define the phase estimation problem in generality.

Let U be an arbitrary unknown unitary operation acting on n qubits. Let $|\psi\rangle$ be an eigenvector of U. But now the eigenvalue is not restricted to being $+1$ or -1. The eigenvalue can be any complex number on the unit circle. So the eigenvalue is of the form $e^{2\pi i\varphi}$, where φ can be any element of the interval $[0, 1) = \{\varphi : 0 \leq \varphi < 1\}$.

Definition 22.1 (Phase estimation problem)**.** In the *phase estimation problem* we are given: a black-box for a multiplicity-controlled-U gate with ℓ control qubits and

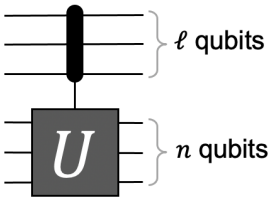

Figure 166: Caption.

one copy of state $|\psi\rangle$, which is an eigenvector of U with eigenvalue $e^{2\pi i\varphi}$ for some $\varphi \in [0, 1)$. The goal is to determine an ℓ-bit approximation of the value of φ. (Since φ can take on a continuum of values, determining it *exactly* is unreasonable.)

There's an *exact case* of this where φ is an ℓ-bit binary fraction. This case is simpler to work with than the general case. An ℓ-bit binary fraction is a rational number with 2^ℓ in the denominator and an integer $a \in \{0, 1, \ldots, 2^\ell - 1\}$ in the numerator. In other words, the binary representation of φ can be written exactly with only ℓ bits occurring after the radix point. For example, for $\ell = 6$,

$$\frac{19}{2^6} = 0.010011. \tag{262}$$

The *general case* is where φ is arbitrary. For example, if $\varphi = \frac{1}{3}$ then

$$\phi = 0.\overline{01} = 0.010101010101\ldots \tag{263}$$

(where the pattern repeats forever). That's not a binary fraction for any ℓ. The 8-bit approximation of this number is 0.01010101 (it's $\frac{1}{3}$ rounded down to the closest

8-bit binary fraction). The 7-bit approximation of this number is 0.0101011 (note that last bit is 1, not 0, because we round *up* to the closest 7-bit binary fraction. We always round φ towards the ℓ-bit binary fraction that's closest. Sometimes that means rounding down and sometimes that means rounding up.

Also, since $[0, 1)$ represents points on the unit circle (where $e^{2\pi i \cdot 1} = e^{2\pi i \cdot 0}$), we define distances within $[0, 1)$ with "wrap-around," representing $\varphi = 1$ as $\varphi = 0$. By this convention, if $1 - \frac{1}{2^{\ell+1}} < \varphi < 1$ then φ rounds *up* to 0.

For our applications, we will want to solve the general case of phase estimation, but it's conceptually useful to to first solve this problem in the exact case.

22.4 Solving the exact case

Here we will solve the phase estimation problem in the exact case—which turns out to be quite easy. In the exact case,

$$\varphi = \frac{a}{2^\ell} = 0.a_1 a_2 \dots a_\ell \,, \tag{264}$$

where $a \in \{0, 1, \dots, 2^\ell - 1\}$. Note that the eigenvalue can be written as

$$e^{2\pi i \varphi} = e^{2\pi i a / 2^\ell} = \left(e^{2\pi i / 2^\ell}\right)^a = \omega^a, \tag{265}$$

where ω is a primitive 2^ℓ-th root of unity. It clarifies things to think of the eigenvalue as ω^a in this manner. Note that the goal of determining the eigenvalue parameter φ is now equivalent to determining the value of the number a.

We'll start by putting the control qubits into a uniform superposition of all computational basis sates on ℓ qubits, which is accomplished by ℓ Hadamard transforms. And then we'll perform the multiplicity-controlled-U gate.

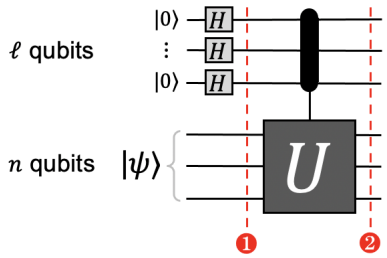

Figure 167: Caption.

Let's determine the output state of this quantum circuit.

State at stage ❶

The $H^{\otimes \ell}$ operation on $|0\rangle^{\otimes \ell}$ produces that state

$$\Big(|0...00\rangle + |0...01\rangle + |0...10\rangle + |0...11\rangle + \cdots + |1...11\rangle\Big)\,|\psi\rangle$$
$$= \Big(|0\rangle + |1\rangle + |2\rangle + |3\rangle + \cdots + \left|2^\ell - 1\right\rangle\Big)\,|\psi\rangle\,. \tag{266}$$

State at stage ❷

The multiplicity-controlled-U gate changes the state to

$$|0\rangle\,|\psi\rangle + |1\rangle\,U\,|\psi\rangle + |2\rangle\,U^2\,|\psi\rangle + |3\rangle\,U^3\,|\psi\rangle + \cdots + \left|2^\ell - 1\right\rangle U^{2^\ell - 1}\,|\psi\rangle$$
$$= |0\rangle\,|\psi\rangle + |1\rangle\,\omega^a\,|\psi\rangle + |2\rangle\,\omega^{2a}\,|\psi\rangle + |3\rangle\,\omega^{3a}\,|\psi\rangle + \cdots + \left|2^\ell - 1\right\rangle \omega^{(2^\ell - 1)a}\,|\psi\rangle \tag{267}$$
$$= \Big(|0\rangle + \omega^a\,|1\rangle + \omega^{2a}\,|2\rangle + \omega^{3a}\,|3\rangle + \cdots + \omega^{(2^\ell - 1)a}\left|2^\ell - 1\right\rangle\Big)\,|\psi\rangle\,. \tag{268}$$

Now, do you recognize this state of the first ℓ qubits? Haven't you seen the state $|0\rangle + \omega^a\,|1\rangle + \omega^{2a}\,|2\rangle + \cdots + \omega^{(2^\ell - 1)a}\left|2^\ell - 1\right\rangle$ before?

It's the Fourier basis state $F_{2^\ell}\,|a\rangle$ (see Eq. (184)). Remember that our goal is to determine a. The fact that the Fourier basis states for all the potential values of parameter a are orthogonal is a good sign. It means that they are distinguishable in principle. But we can also explicitly compute the value of a using a polynomial number of gates. Can you see how?

We can compute a by applying the the *inverse Fourier transform*. If we apply $F^*_{2^\ell}$ to $F_{2^\ell}\,|a\rangle$, the result is $|a\rangle$. So our phase estimation algorithm for the exact case is the quantum circuit in figure 168.

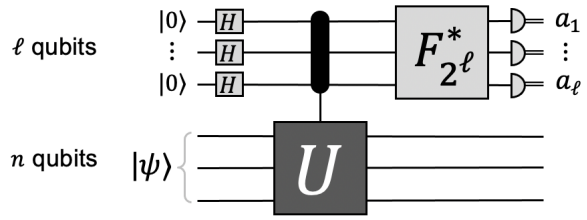

Figure 168: Quantum algorithm for phase estimation in the exact case.

The output of the ℓ measured qubits is a, the numerator of the binary fraction for φ, or, equivalently, the ℓ bits of the binary representation of φ.

Note that the algorithm makes only one query to the multiplicity-controlled-U gate, and all the other operations in the algorithm (the grey boxes) can be implemented with a polynomial number of elementary gates (with respect to ℓ, the number of control qubits). The dominant cost is that of computing $F^*_{2^\ell}$, which costs $O(\ell^2)$ elementary gates (using the method of section 19.3).

By the way, the $\ell = 1$ version of the exact case is that simple example that we solved in section 22.1 (and in that case $\omega = -1$).

22.5 Solving the general case

Now, what happens if we use this same algorithm for the general case, where φ can be *any* real number between 0 and 1? Recall that, in that case, we need to determine the ℓ-bit binary number that best approximates the true value of φ. We can write φ as an ℓ-bit binary fraction plus a quantity δ that represents the remaining bits that get rounded up[33] or down. More precisely,

$$\varphi = \frac{a}{2^\ell} + \delta, \tag{269}$$

where $a \in \{0, 1, \ldots, 2^\ell - 1\}$ and

$$|\delta| \leq \frac{1}{2^{\ell+1}}. \tag{270}$$

If φ gets rounded down then δ is positive; if φ gets rounded up then δ is negative. If $\delta = 0$, we are in the exact case. Let's analyze what the circuit that we used for the exact case does in the case where $\delta \neq 0$.

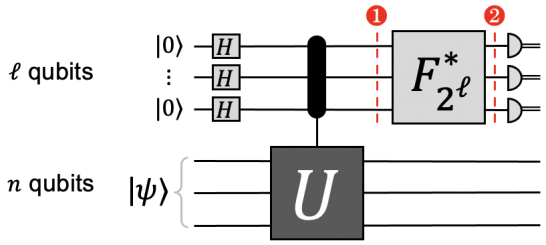

Figure 169: Quantum algorithm for phase estimation applied in the general case.

Since $|\psi\rangle$ is an eigenvector of U, the state of the second register (the last n qubits) does not change. All the action is with the first register (the first ℓ qubits). We trace through the evolution of the first ℓ qubits.

[33] In the special case of "wrap around," $\frac{1}{2^{\ell+1}} + \delta = 1$ (since we represent $\varphi = 1$ as $\varphi = 0$).

State at stage ❶

After the multiplicity-controlled-U gate, the state of the first ℓ qubits is

$$\frac{1}{\sqrt{2^\ell}} \sum_{b=0}^{2^\ell-1} \left(e^{2\pi i\varphi}\right)^b |b\rangle = \frac{1}{\sqrt{2^\ell}} \sum_{b=0}^{2^\ell-1} e^{2\pi i\left(\frac{a}{2^\ell}+\delta\right)b} |b\rangle \tag{271}$$

$$= \frac{1}{\sqrt{2^\ell}} \sum_{b=0}^{2^\ell-1} \omega^{ab} e^{2\pi i\delta b} |b\rangle\,, \tag{272}$$

where $\omega = e^{2\pi i/2^\ell}$. The effect of δ is highlighted in red. If $\delta = 0$ then $e^{2\pi i\delta b} = 1$ and the expression is $F_{2^\ell} |a\rangle$, which is consistent with what we obtained for the exact case.

State at stage ❷

After applying the inverse Fourier transform, the state becomes

$$F_{2^\ell}^* \left(\frac{1}{\sqrt{2^\ell}} \sum_{b=0}^{2^\ell-1} \omega^{ab} e^{2\pi i\delta b} |b\rangle \right) = \frac{1}{\sqrt{2^\ell}} \sum_{b=0}^{2^\ell-1} \omega^{ab} e^{2\pi i\delta b} \left(\frac{1}{\sqrt{2^\ell}} \sum_{c=0}^{2^\ell-1} \omega^{-bc} |c\rangle \right) \tag{273}$$

$$= \frac{1}{2^\ell} \sum_{c=0}^{2^\ell-1} \left(\sum_{b=0}^{2^\ell-1} \omega^{b(a-c)} e^{2\pi i\delta b} \right) |c\rangle\,. \tag{274}$$

Now, recall that the correct outcome for the phase estimation problem is a (the bits of an ℓ-bit approximation of φ). What's the probability that measuring the state in Eq. (274) results in outcome a? To figure this out, we look at the amplitude of the $|a\rangle$ term in this expression. Notice that if we substitute a for c then the factor $\omega^{b(a-c)}$ simplifies to $\omega^0 = 1$. So the amplitude of the $|a\rangle$ term in Eq. (274) is the sum

$$\frac{1}{2^\ell} \sum_{b=0}^{2^\ell-1} e^{2\pi i\delta b}. \tag{275}$$

As a reality check, what is this sum in the exact case, where $\delta = 0$? Can you see why it's 1? This makes sense, because, for the exact case, we've already seen that the measurement outcome is a with probability 1.

Returning to our case of interest, where $0 \neq |\delta| \leq 1/2^{\ell+1}$, we're interested in the absolute value squared of the amplitude in Eq. (275). At first glance, the sum may look a little daunting, but there's a closed-form expression for it. Can you see what it is?

The sum in Eq. (275) is a geometric series.[34] Therefore, we can use the formula for the sum of a geometric series to obtain

$$\frac{1}{2^\ell}\sum_{b=0}^{2^\ell-1} e^{2\pi i\delta b} = \frac{1}{2^\ell}\frac{1-(e^{2\pi i\delta})^{2^\ell}}{1-e^{2\pi i\delta}}. \tag{276}$$

What we're going to do next is use some simple geometric reasoning to show that the absolute value of this expression is at least $\frac{2}{\pi}$.

Lemma 22.1. *If δ is such that $0 \neq |\delta| \leq 1/2^{\ell+1}$ then*

$$\frac{1}{2^\ell}\left|\frac{1-(e^{2\pi i\delta})^{2^\ell}}{1-e^{2\pi i\delta}}\right| \geq \frac{2}{\pi}. \tag{277}$$

Proof. To lower-bound the expression, we'll show that the numerator is not too small and the denominator is not too large. We give the proof for the case where $\delta > 0$ (the case where $\delta < 0$ is similar, with upside-down versions of figures 170 and 171).

First, let's show that the denominator is not too large. Figure 170 shows 1 and $e^{2\pi i\delta}$ as points in the complex plane.

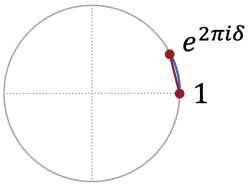

Figure 170: Geometric view of $|1-e^{2\pi i\delta}|$ as the distance between two points in $\mathbb{C}$.

The length of the red line segment is $|1-e^{2\pi i\delta}|$. The length of the red line is upper bounded by the arc-length between the two points, which is $2\pi\delta$. Therefore, we can upper bound of the denominator as

$$|1-e^{2\pi i\delta}| \leq 2\pi\delta. \tag{278}$$

Next, we'll lower bound the numerator. In this case, $|1-e^{2\pi i\delta 2^\ell}|$ is the length of the red line in figure 171.

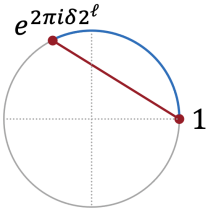

Figure 171: Geometric view of $|1-e^{2\pi i\delta 2^\ell}|$ as the distance between two points in $\mathbb{C}$.

[34] For $s \neq 1$, the formula for the sum of the geometric series $1+s+s^2+\cdots+s^{m-1} = \frac{1-s^m}{1-s}$.

Let's also consider the arc length between the two points, which is $2\pi\delta 2^\ell$. Note that the arc-length is *not* a lower bound of the distance—it's longer than the line segment. Think of the arc-length as the length of the line times some stretch-factor. How big is the stretch-factor? If the line segment is short then the stretch-factor is just slightly larger than 1 (in that case, the line and arc have almost the same lengths).

But the line and arc need not be short. The extremal case is where δ is as large as possible: $\delta = 1/2^{\ell+1}$. In that case, $(e^{2\pi i\delta})^{2^\ell} = e^{\pi i} = -1$, so the length of the line is 2 and the length of the arc is π. Thus, the maximum possible stretch-factor is $\pi/2$.

Therefore, we can lower bound the length of the line by the arc-length times $2/\pi$. Multiplying the arc-length by the reciprocal of the maximum stretch-factor compensates for how much longer the arc-length can be than the line segment. Therefore, the numerator is lower bounded as

$$|1 - e^{2\pi i\delta 2^\ell}| \geq 2\pi\delta 2^\ell\left(\frac{2}{\pi}\right) = 4\delta 2^\ell. \tag{279}$$

Now, we can substitute in our bounds in Eq. (278) and Eq. (279) for the numerator and the denominator to obtain

$$\frac{1}{2^\ell}\left|\frac{1-(e^{2\pi i\delta})^{2^\ell}}{1-e^{2\pi i\delta}}\right| \geq \frac{1}{2^\ell}\frac{4\delta 2^\ell}{2\pi\delta} = \frac{2}{\pi}. \tag{280}$$

■

This means that, if we measure (in the computational basis), we get the correct answer a with probability at least $4/\pi^2 = 0.405...$ (squaring the amplitude). That's slightly more than 40%. This might not seem like a very high success probability. But, for our purposes, what's important is that it's a *constant* (independent of ℓ and n). In the contexts where we will use phase estimation to achieve something else (such as to factor a number), we will be able to amplify the success probability by repeating the entire process a few times.

In summary, for the phase estimation problem, you are given an multiplicity-controlled-U gate with ℓ control-qubits and a state $|\psi\rangle$ that's an eigenvector of U with eigenvalue $e^{2\pi i\varphi}$. Your goal is to find an ℓ-bit approximation of φ. The following quantum circuit solves this problem with success probability at least $\frac{4}{\pi^2} = 0.405\ldots$.

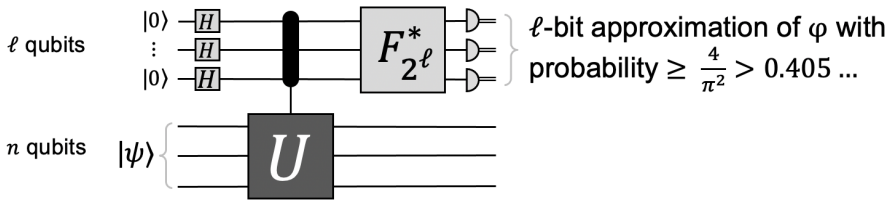

Figure 172: Summary of phase-estimation algorithm for approximating the eigenvalue $e^{2\pi i\varphi}$ of $|\psi\rangle$.

The algorithm makes one query to the multiplicity-controlled-U gate and has $O(\ell^2)$ elementary gates (the dominant cost being the implementation of $F^*_{2^\ell}$).

22.6 The case of superpositions of eigenvectors

Before ending this section, I'd like to bring your attention to a slightly different scenario. Suppose that, instead of being given an eigenvector of U, we're given a superposition of two eigenvectors with different eigenvalues.

Suppose that we're provided with a state of the form $\alpha_1 |\psi_1\rangle + \alpha_2 |\psi_2\rangle$, where:

- $|\psi_1\rangle$ is an eigenvector of U with eigenvalue $e^{2\pi i\varphi_1}$,
- $|\psi_2\rangle$ is an eigenvector of U with eigenvalue $e^{2\pi i\varphi_2}$,
- $|\alpha_1|^2 + |\alpha_2|^2 = 1$, and $\varphi_1 \neq \varphi_2$.

What happens if we run our phase estimation algorithm with this state in the target register?

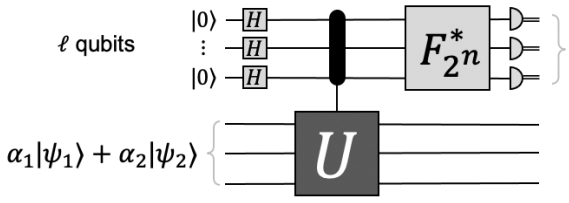

Figure 173: Scenario where input to phase algorithm is a superposition of two eigenvectors.

Exercise 22.1. For the exact case, where φ_1 and φ_2 are both ℓ-bit binary integers, show that the output of the circuit in figure 173 is

$$\begin{cases} \varphi_1 & \text{with probability } |\alpha_1|^2 \\ \varphi_2 & \text{with probability } |\alpha_2|^2. \end{cases} \tag{281}$$

So the net result is exactly the same as what occurs if, instead of a superposition of eigenvalues, we had just randomly selected one of the eigenvectors as

$$\begin{cases} |\psi_1\rangle & \text{with probability } |\alpha_1|^2 \\ |\psi_2\rangle & \text{with probability } |\alpha_2|^2 \end{cases} \tag{282}$$

and then input the selected eigenvector to the target register.

That's for the exact case. For the general case, it's similar, with the statement of the result qualified as "with success probability at least $4/\pi^2$." Also, although I've described the case of a superposition of *two* eigenvectors, there's nothing so special about two. There are similar results for the case of superpositions of more than two eigenvectors.

23 The order-finding problem

23.1 Greatest common divisors and Euclid's algorithm

In section 24, I will show you quantum algorithms for two problems: one is called the *order-finding problem* and the other is the *factoring problem*. I will show you how these algorithms can be based on the framework of the phase estimation problem that we saw in section 22.

In the present section, I will review some basics about divisors, common divisors, and the greatest common divisor of two integers.

Of course, we know that factoring a number x (a positive integer) is about finding its divisors (where the divisors are the numbers that divide x). If we have two numbers, x and y, then the *common divisors* of x and y are the numbers that divide both of them. For example, for the numbers 28 and 42 the common divisors are: 1, 2, 7, and 14. (4 is not a common divisor because, although 4 divides 28, it does not divide 42.)

Definition 23.1 (greatest common divisor (GCD))**.** For two numbers x and y, their *greatest common divisor* is the largest number that divides both x and y. This is commonly denoted as $\gcd(x, y)$.

The GCD of 28 and 42 is 14. What is the GCD of 16 and 21?
The answer is 1, since there is no larger integer that divides both of them.

Definition 23.2 (relatively prime)**.** We say that numbers x and y are *relatively prime* if $\gcd(x, y) = 1$. That is, they have no common divisors, except the trivial divisor 1.

Note that 16 and 21 are not prime but they are relatively prime.

Superficially, the problem of finding common divisors resembles the problem of finding divisors—which is the factoring problem. If x and y are n-bit numbers then a brute-force search would have to check among exponentially many possibilities.

In fact, the problem of finding greatest common divisors is considerably easier than factoring. It has been known for over 2,000 years that there is an efficient algorithm for computing GCDs. That's how long ago Euclid published his algorithm[35] (now known as the *Euclidean algorithm*). Of course, there were no computers back then, but Euclid's algorithm could be carried out by a hand calculation and it requires

[35]Euclid (c. 320–275 BC). See: T. L. Heath (1925). *The thirteen books of Euclid's Elements*, Vol. 1. Dover publications Inc., New York, Second Edition.

$O(n)$ arithmetic operations for n-digit numbers. This translates into a gate cost of $O(n^2 \log n)$. Let's remember that Euclid's GCD algorithm exists, because we're going to make use of it.

Now, I'd like to briefly review a few more properties of numbers. Let $m \geq 2$ be an integer that's not necessarily prime. Recall that $\mathbb{Z}_m = \{0, 1, 2, \ldots, m-1\}$ and

$$\mathbb{Z}_m^* = \Big\{ x \in \mathbb{Z}_m : x \text{ has a multiplicative inverse mod } m \Big\}, \tag{283}$$

where the latter is a group, with respect to multiplication mod m. For example,

$$\mathbb{Z}_{21}^* = \{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}. \tag{284}$$

It turns out that x has a multiplicative inverse mod m if and only if $\gcd(x, m) = 1$. We do not prove this here, but you can see that 3, 6, and 7, and the other numbers in $\mathbb{Z}_{21}$ that are excluded from $\mathbb{Z}_{21}^*$ are exactly those that have non-trivial common factors with 21.

So we have an equivalent way of defining $\mathbb{Z}_m^*$ as

$$\mathbb{Z}_m^* = \Big\{ x \in \mathbb{Z}_m : \mathsf{GCD}(x, m) = 1 \Big\}. \tag{285}$$

Now, let's take an element of $\mathbb{Z}_{21}^*$ and list all of its powers. Suppose we pick 5. The powers of 5 (5^0, 5^1, 5^2, ...) are

$$1, 5, 4, 20, 16, 17, 1, 5, 4, 20, 16, 17, 1, 5, 4, 20, 16, \ldots \tag{286}$$

Notice that it's periodic ($5^6 = 1$, and then the subsequent powers repeat the same sequence). The *period* of the sequence is 6.

If we were to pick 4 instead of 5 then the period of the sequence is different. The powers of 4 are:

$$1, 4, 16, 1, 4, 16, \ldots \tag{287}$$

so the period is 3.

Definition 23.3 (order of $a \in \mathbb{Z}_m^*$)**.** For any $a \in \mathbb{Z}_m^*$ define *the order of a modulo m (denoted as $ord_m(a)$)* as the minimum positive r such that $a^r \equiv 1 \pmod{m}$. This is the period of the sequence of powers of a.

Related to all this, we can define the computational problem of determining $\mathrm{ord}_m(a)$ from m and a. This problem has an interesting relationship to the factoring problem.

23.2 The order-finding problem and its relation to factoring

Definition 23.4 (order-finding problem)**.** The input to the order-finding problem is two n-bit integers m and a, where $m \geq 2$ and $a \in \mathbb{Z}_m^*$. The output is $ord_m(a)$, the order of a modulo m.

A brute-force algorithm for order-finding is to compute the sequence of powers of a $(a, a^2, a^3, \dots)$ until a 1 is reached. That can take exponential-time when the modulus is an n-bit number because the order can be exponential in n. Each power of a can be computed efficiently by the repeated-squaring trick, but there are potentially exponentially many different powers of a to check. In fact, there is no known polynomial-time *classical* algorithm for the order-finding problem.

But why should we care about solving the order-finding problem efficiently? It might come across as an esoteric problem in computational number theory. One reason to care is that the factoring problem *reduces* to the order-finding problem. If we can solve the order finding-problem efficiently then we can use that to factor numbers efficiently.

What I'm going to do next is show how any efficient algorithm (classical or quantum) for the order-finding problem can be turned into an efficient algorithm for factoring. This is true for the general factoring problem; however, for simplicity, I'm only going to explain it for the case of factoring numbers that are the product of two distinct primes. That's the hardest case, and the case that occurs in cryptographic applications.

Let our input to the factoring problem be an n-bit number m, such that $m = pq$, where p and q are distinct primes. Our goal is to determine p and q, with a gate-cost that is polynomial in n.

The first step is to select an $a \in \mathbb{Z}_m^*$ and use our quantum order-finding algorithm[36] as a subroutine to compute $r = \text{ord}_m(a)$. Note that, since $a^r \equiv 1 \pmod{m}$, it holds that $a^r - 1$ is a multiple of m. In other words, m divides $a^r - 1$.

Now, the order r is either an even number or an odd number (it can go either way, depending on which a we pick). Something interesting happens when r is an even number. If r is even then a^r is a square, with square root $a^{r/2}$, and we can express $a^r - 1$ using the standard formula for a difference-of-squares as

$$a^r - 1 = (a^{r/2} + 1)(a^{r/2} - 1). \tag{288}$$

[36] In the next section, we'll see that there is an efficient quantum algorithm for the order-finding problem that we can use for this.

Since m divides $a^r - 1$ and $m = pq$, it follows that p and q each divide $a^r - 1$. It's well known that if a prime divides a product of two numbers then it must divide one of them. Also, it's not possible that both p and q divide the factor $a^{r/2} - 1$. Why not? Because that would contradict the fact that r is, by definition, the *smallest* number for which $a^r \equiv 1 \pmod{m}$. Therefore, there are three possibilities for the factors that p and q divide:

1. p divides $a^{r/2} + 1$ and q divides $a^{r/2} - 1$.
2. q divides $a^{r/2} + 1$ and p divides $a^{r/2} - 1$.
3. p and q both divide $a^{r/2} + 1$.

In the first two cases, p and q divide different factors; in the third case, they divide the same factor. Our reduction works well when r is even *and* p and q divide different factors. With this in mind, let's define a to be "lucky" when these conditions occur.

Definition 23.5 (of a *lucky* $a \in \mathbb{Z}_m^*$)**.** With respect to some $m \geq 2$, define an $a \in \mathbb{Z}_m^*$ to be *lucky* if $r = \mathrm{ord}_m(a)$ is an even number and m does not divide $a^{r/2} + 1$.

Given an $a \in \mathbb{Z}_m^*$ that is lucky in the above sense, it holds that

$$\gcd(a^{r/2} + 1, m) \in \{p, q\}. \tag{289}$$

This enables us to easily factor m by computing $\gcd(a^{r/2} + 1, m)$. The repeated-squaring trick enables us to compute $a^{r/2}$ with a gate cost of $O(n^2 \log n)$ and Euclid's algorithm enables us to compute $\gcd(a^{r/2} + 1, m)$ with a gate cost of $O(n^2 \log n)$.

The outstanding matter is to find a lucky $a \in \mathbb{Z}_m^*$. How do we do that? Unfortunately, there is no efficient *deterministic* method known for finding a lucky $a \in \mathbb{Z}_m^*$. However, it's known that, for any $m \geq 2$, the number of lucky $a \in \mathbb{Z}_m^*$ is quite large.

Lemma 23.1. *For all $m = pq$, where p and q are distinct primes, at least half of the elements of $\mathbb{Z}_m^*$ are lucky.*

So what we can do is *randomly* select an $a \in \mathbb{Z}_m^*$. The selection will be lucky with probability at least $\frac{1}{2}$, and therefore the resulting procedure produces a factors of m with probability at least $\frac{1}{2}$. And we can check whether the output of the procedure is a factor of m. We can repeat the process several times to boost the success probability.

So that's how an efficient algorithm for the order-finding problem can be used to factor a number efficiently. Now we can focus our attention on finding an efficient algorithm for order-finding.

24 Shor's algorithm for order-finding

In this section, I will first show you an efficient quantum algorithm for the order-finding problem. And then, in section 25, I will show you an efficient algorithm for factoring (using the reduction to the order-finding algorithm from section 23.2).

Let's begin with an input instance to the order-finding problem, m and a, which are both n-bit numbers and for which $a \in \mathbb{Z}_m^*$. The goal of the algorithm is to determine $r = \mathrm{ord}_m(a)$.

24.1 Order-finding in the phase estimation framework

Our approach makes use of the framework of the phase-estimation algorithm (explained in section 22). Recall that, for this framework, we need a unitary operation and an eigenvector. Our unitary operation is $U_{a,m}$ defined such that, for all $b \in \mathbb{Z}_m^*$,

$$U_{a,m} |b\rangle = |ab\rangle \tag{290}$$

(where by ab we mean the product of a and b modulo m). As for the eigenvector, we'll begin with

$$|\psi_1\rangle = \tfrac{1}{\sqrt{r}} \Big(|1\rangle + \omega^{-1} |a\rangle + \omega^{-2} |a^2\rangle + \cdots + \omega^{-(r-1)} |a^{r-1}\rangle \Big) \tag{291}$$

$$= \tfrac{1}{\sqrt{r}} \sum_{j=0}^{r-1} \omega^{-j} \left| a^j \right\rangle , \tag{292}$$

where $\omega = e^{2\pi i (1/r)}$ (a primitive r-th root of unity).

To see why $|\psi_1\rangle$ is an eigenvector of $U_{a,m}$, consider what happens if we apply $U_{a,m}$ to $|\psi_1\rangle$. Note that $U_{a,m}$ maps each $|a^k\rangle$ to $|a^{k+1}\rangle$, where $|a^r\rangle = |1\rangle$. Therefore,

$$U_{a,m} |\psi_1\rangle = \tfrac{1}{\sqrt{r}} \sum_{j=0}^{r-1} \omega^{-j} \left| a^{j+1} \right\rangle \tag{293}$$

$$= \tfrac{1}{\sqrt{r}} \sum_{j=0}^{r-1} \omega \, \omega^{-(j+1)} \left| a^{j+1} \right\rangle \tag{294}$$

$$= \omega \, |\psi_1\rangle . \tag{295}$$

Therefore $|\psi_1\rangle$ is an eigenvector of $U_{a,m}$ with eigenvalue $\omega = e^{2\pi i (1/r)}$. What's interesting about this is that, if we can estimate the eigenvalue's parameter $1/r$ with sufficiently many bits of precision, then we can deduce what r is. In order to do this using the phase estimation algorithm, we need to efficiently simulate a multiplicity-controlled-$U_{a,m}$ gate and also to construct the state $|\psi_1\rangle$.

24.1.1 Multiplicity-controlled-$U_{a,m}$ gate

Here I will show you a rough sketch of how to efficiently compute a multiplicity-controlled-$U_{a,m}$ gate. Consider the mapping of the multiplicity-controlled-$U_{a,m}$ gate with ℓ control qubits.

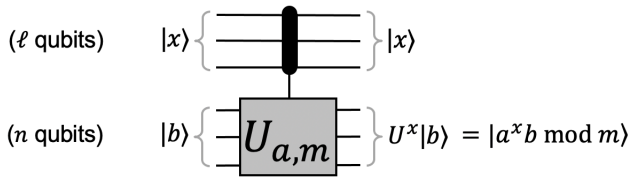

Figure 174: Multiplicity-controlled-$U_{a,m}$ gate.

For each $x \in \{0,1\}^\ell \equiv \{0,1,\ldots,2^\ell - 1\}$ and $b \in \mathbb{Z}_m^*$, this gate maps $|x\rangle\,|b\rangle$ to

$$|x\rangle\,(U_{a,m})^x\,|b\rangle = |x\rangle\,|a^x b\rangle \tag{296}$$

(which is equivalent to $U_{a,m}$ being applied x times).

We can compute this efficiently(with respect to ℓ and n) by using the repeated squaring trick to exponentiate a. The number of gates is $O(\ell n \log n)$.

A word of caution: Note that, *in general*, if we can efficiently compute some unitary U then it does *not* always follow that we can compute a multiplicitly-controlled-U gate efficiently. The $U_{a,m}$ that arises here has special structure that permits this.

24.1.2 Precision needed to determine $1/r$

Now, how many bits of precision ℓ should we use in our phase estimation of $\frac{1}{r}$? It should be sufficient to uniquely determine $\frac{1}{r}$. We know that $r \in \{1,2,3,\ldots,m\}$ and the corresponding reciprocals are $\left\{\frac{1}{1},\frac{1}{2},\frac{1}{3},\ldots,\frac{1}{m}\right\}$. The positions of these reciprocals on the real line are shown in figure 175.

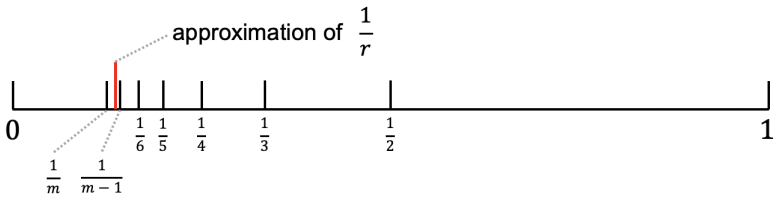

Figure 175: The positions of $\left\{\frac{1}{1},\frac{1}{2},\frac{1}{3},\ldots,\frac{1}{m}\right\}$ on the real line.

The smallest gap occurs between $\frac{1}{m}$ and $\frac{1}{m-1}$. The size of this gap is

$$\frac{1}{m-1} - \frac{1}{m} = \frac{m-(m-1)}{m(m-1)} = \frac{1}{m(m-1)} \approx \frac{1}{m^2}. \tag{297}$$

This means the approximation must be within $\frac{1}{2}\frac{1}{m^2}$ of $\frac{1}{r}$. Since m is an n-bit number, $m \approx 2^n$ and $\frac{1}{2}\frac{1}{m^2} \approx \frac{1}{2^{2n+1}}$ which corresponds to setting $\ell = 2n+1$.

24.1.3 Can we construct $|\psi_1\rangle$?

So far so good. But to efficiently implement the phase estimation algorithm, we also need to be able to efficiently create the eigenvector $|\psi_1\rangle$. Of course, if we already know what r is, then this is straightforward. But the algorithm does not know r at this stage; r is what the algorithm is trying to determine.

It seems very hard to construct this state without knowing what r is. This is a serious problem; it's not known how to construct this state directly. Instead of producing that state $|\psi_1\rangle$, our way forward will be to use a different state.

24.2 Order-finding using a random eigenvector $|\psi_k\rangle$

Let's consider alternatives to the eigenvector $|\psi_1\rangle = \frac{1}{\sqrt{r}} \sum_{j=0}^{r-1} \omega^{-j} \left|a^j\right\rangle$ (with $\omega = e^{2\pi(\frac{1}{r})}$). Other eigenvectors of $U_{a,m}$ are

$$|\psi_2\rangle = \tfrac{1}{\sqrt{r}} \sum_{j=0}^{r-1} \omega^{-2j} \left|a^j\right\rangle \tag{298}$$

$$\vdots \qquad \vdots$$

$$|\psi_k\rangle = \tfrac{1}{\sqrt{r}} \sum_{j=0}^{r-1} \omega^{-kj} \left|a^j\right\rangle \tag{299}$$

$$\vdots \qquad \vdots$$

$$|\psi_r\rangle = \tfrac{1}{\sqrt{r}} \sum_{j=0}^{r-1} \omega^{-rj} \left|a^j\right\rangle. \tag{300}$$

For each $k \in \{1, 2, \ldots, r\}$, the eigenvalue of $|\psi_k\rangle$ is $e^{2\pi(\frac{k}{r})}$.

Suppose that we are able to devise an efficient procedure that somehow produces a random sample from the r different eigenvectors (where each $|\psi_k\rangle$ occurrs with probability $\frac{1}{r}$). Can we deduce r from this? There are two versions of this scenario, depending on whether or not the procedure reveals k.

24.2.1 When k is known

First, let's suppose that the output of the procedure is the pair $(k, |\psi_k\rangle)$, with each possibility occurring with probability $\frac{1}{r}$. In this case, we can use the phase estimation algorithm (with the random $|\psi_k\rangle$) to get an approximation of $\frac{k}{r}$ and then divide the approximation by k to turn it into an approximation of $\frac{1}{r}$, and proceed from there as with the case of $|\psi_1\rangle$. The details of this case are straightforward.

24.2.2 When k is not known

Now, let's change the scenario slightly and assume that we have an efficient procedure that somehow generates a random $|\psi_k\rangle$ (uniformly distributed among the r possibilities) as output—but without revealing k. Can we still extract r from $|\psi_k\rangle$ alone?

Using the phase estimation algorithm, we can obtain a binary fraction $0.b_1b_2\ldots b_\ell$ that estimates $\frac{k}{r}$ within ℓ-bits of precision (where we can set the value of ℓ). But how can we determine k and r from this? This is a tricky problem, because we don't know what k to divide by in order to turn an approximation of $\frac{k}{r}$ into an approximation of $\frac{1}{r}$. But there is a way to do this using the fact that we have an upper bound of the denominator: r cannot be larger than the modulus m.

Let's carefully restate the situation as follows. We have an n-bit integer m and we are also given an ℓ-bit approximation $0.b_1b_2\ldots b_\ell$ of one of the elements of

$$\left\{ \frac{k}{r} \;\middle|\; \text{where } r \in \{1, 2, \ldots, m\} \text{ and } k \in \{1, 2, \ldots, r\} \right\}. \tag{301}$$

Our goal is to determine which element of the set is being approximated.

Consider the example where the upper bound on the denominator is 8 (i.e., $m = 8$). Figure 176 shows *all the candidates* for $\frac{k}{r}$.

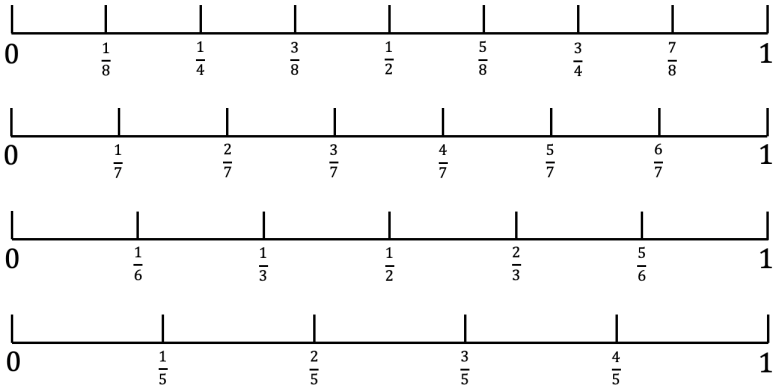

Figure 176: The set of candidates for $\frac{k}{r}$ in the $m = 8$ case.

Note that denominators 2, 3, and 4 are covered by the cases of 6 and 8 in the denominator. Figure 177 shows what all these candidates look like when they're combined on one line.

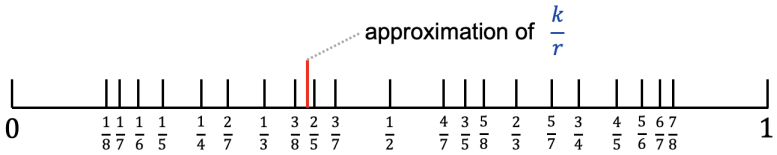

Figure 177: The set of candidates for $\frac{k}{r}$ in the $m = 8$ case (shown combined).

Now, suppose that we are able to obtain a 7-bit approximation of one of these candidates. Say it's $0.0110011 = 51/2^7$. This number is exactly $51/128$, but that's not one of the candidates (because the denominator is larger than 8). The rational number $\frac{2}{5}$ turns out to be the *unique* candidate within distance $\frac{1}{2^7}$ from 0.0110011.

In general, how good an approximation do we need in order to single out a unique $\frac{k}{r}$? This depends on the smallest gap among the set of candidates. It can be shown that the gap is smallest at the ends: $\frac{1}{m-1} - \frac{1}{m} \approx \frac{1}{m^2}$. Therefore, setting $\ell = 2n + 1$ provides the correct level of precision to guarantee that there is a unique rational number $\frac{k}{r}$ where $r \leq m$ that's close to the approximation.

But how do we find the rational number? If m is an n-bit number then the number of candidates (i.e., what figure 177 shows for the $m = 8$ case) is exponential in n. So a brute-force search among all the candidates is not going to be efficient.

It turns out that there is a known efficient algorithm that's tailor-made for this problem, called the *continued fractions algorithm*.[37] The continued-fractions algorithm enables us to determine k and r efficiently from a $(2n + 1)$-bit approximation of $\frac{k}{r}$.

But perhaps you've already noticed that there is a major problem with all this: that of *reduced fractions*.

24.2.3 A snag: reduced fractions

The problem of reduced fractions is that different k and r can correspond to exactly the same number. For example, if $k = 34$ and $r = 51$ then $\frac{k}{r} = \frac{34}{51} = \frac{2}{3}$. There is no way to distinguish between $\frac{34}{51}$ and $\frac{2}{3}$. The continued fractions algorithm only provides

[37] The continued fractions algorithm was discovered in the 1600s by Christiaan Huygens, a Dutch physicist, astronomer, and mathematician who made several amazing contributions to diverse fields (for example, he discovered Saturn's rings, and he contributed to the theory of how light propagates).

a k and r in reduced form, which does not necessarily correspond to the actual r. If it should happen that k has the property that $\gcd(k, r) = 1$ then this problem does not occur. In that case, the fraction is already in reduced form.

Recall that, in our setting, we are assuming that k is uniformly sampled from the set $\{1, 2, \ldots, r\}$. What can we say about the probability of such a random k being relatively prime to r? This probability is the ratio of the size of the set $\mathbb{Z}_r^*$ to the set $\mathbb{Z}_r$. The ratio need not be constant, but is not too small. It is known to be at least $\Omega(1/\log\log r)$ in size. When the modulus is n-bits, this is at least $\Omega(1/\log n)$.

This means that $O(\log n)$ repetitions of the process is sufficient for the probability of a k that's relatively prime to r to arise with constant probability. Procedurally, in the case where $\gcd(k, r) > 1$, the result is an r that's smaller than the order (which can be detected because in such cases $a^r \bmod m \neq 1$). The $O(\log n)$ repetitions just introduces an extra log factor into the gate cost.

In fact, we can do better than that. If we make two repetitions then there are two rational numbers $\frac{k_1}{r}$ and $\frac{k_2}{r}$ (where r is the order, which is unknown to us). Call the reduced fractions that we get from the continued fractions algorithm $\frac{k_1'}{r_1'}$ and $\frac{k_2'}{r_2'}$ (where r is a multiple of r_1' and of r_2'). It turns out that, with constant probability, the *least common multiple* of r_1' and r_2' is r. So, with just *two* repetitions of the phase estimation algorithm, we can obtain the order r with constant success probability (independent of n), assuming we use an independent random eigenvector in each run.

24.2.4 Conclusion of order-finding with a random eigenvector

Now, lets step back and see where we are. We're trying to solve the order-finding problem using the phase-estimation algorithm.

We have a multiplicity-controlled-$U_{a,m}$ gate that's straightforward to compute efficiently.

Regarding the eigenvector, *if* we could construct the eigenvector state $|\psi_1\rangle$ *then* we could determine the order r from that. Unfortunately, we don't know how to generate the state $|\psi_1\rangle$ efficiently. We also saw that: if we could generate a randomly sampled pair $(k, |\psi_k\rangle)$ then we could also determine r from that. Unfortunately, we don't know how to generate such a random pair efficiently.

Next, we saw that: if we could generate a randomly sampled $|\psi_k\rangle$ (without the k) then we could still determine r from that. In that case, we had to do considerably more work. But, using the continued-fractions algorithm and some probabilistic analysis, we could make this work. So ... can we generate a random $|\psi_k\rangle$? Unfortunately, we

don't know how to efficiently generate such a random $|\psi_k\rangle$ either.

Are we at a dead end? No, in fact we're almost at a solution! What I'll show next is that if, instead of generating a random $|\psi_k\rangle$ (with probability $\frac{1}{r}$ for each k), we can instead generate a *superposition* of the $|\psi_k\rangle$ (with amplitude $\frac{1}{\sqrt{r}}$ for each k) then we can determine r from this. And this is a state that we *can* generate efficiently.

24.3 Order-finding using a superposition of eigenvectors

The phase estimation algorithm was explained in section 22. Remember, at the very end, in subsection 22.6, I discussed what happens if the target register is in a superposition of eigenvectors? In that case, the outcome is an approximation of the phase of a random eigenvector of the superposition, with probability the amplitude squared. Therefore, setting the target register to state

$$\frac{1}{\sqrt{r}} \sum_{k=1}^{r} |\psi_k\rangle \tag{302}$$

results in the same outcome that occurs when we use a randomly generated eigenvector state—which is the case that we previously analyzed in section 24.2. So we have an efficient algorithm for order-finding using the superposition state in Eq. (302) in place of an eigenvector (it succeeds with constant probability, independent of n).

24.4 Order-finding without requiring an eigenvector

How can we efficiently generate the superposition state in Eq. (302)? In fact, this is extremely easy to do. To see how, expand the state as

$$\begin{aligned}
\frac{1}{\sqrt{r}} \sum_{k=1}^{r} |\psi_k\rangle \;=\; & \frac{1}{r}\Big(|1\rangle \;+\; \omega\,|a\rangle \;+\; \omega^2\,\big|a^2\big\rangle \;+\; \cdots \;+\; \omega^{r-1}\big|a^{r-1}\big\rangle\Big) \\
& + \frac{1}{r}\Big(|1\rangle \;+\; \omega^2\,|a\rangle \;+\; \omega^4\,\big|a^2\big\rangle + \cdots + \omega^{2(r-1)}\big|a^{r-1}\big\rangle\Big) \\
& + \frac{1}{r}\Big(|1\rangle \;+\; \omega^3\,|a\rangle \;+\; \omega^6\,\big|a^2\big\rangle + \cdots + \omega^{3(r-1)}\big|a^{r-1}\big\rangle\Big) \\
& \quad\vdots \qquad\quad \vdots \qquad\quad \vdots \qquad\qquad\quad \vdots \\
& + \frac{1}{r}\Big(|1\rangle \;+\; \omega^r\,|a\rangle + \omega^{2r}\,\big|a^2\big\rangle + \cdots + \omega^{r(r-1)}\big|a^{r-1}\big\rangle\Big) \qquad (303) \\
= \; & |1\rangle \qquad (304)
\end{aligned}$$

where the simplification to $|1\rangle$ is a consequence of $\omega^r = 1$ and the fact that, for all $k \in \{1, 2, \ldots, r-1\}$, it holds that $1 + \omega^k + \omega^{2k} + \cdots + \omega^{(r-1)k} = 0$.

So we're left with $|1\rangle$. The n-bit binary representation of the number 1 is 00...01. So the superposition of eigenvectors in Eq. (302) is merely the computational basis state $|00...01\rangle$, which is indeed trivial to construct. The quantum part of the order-finding algorithm looks like this.

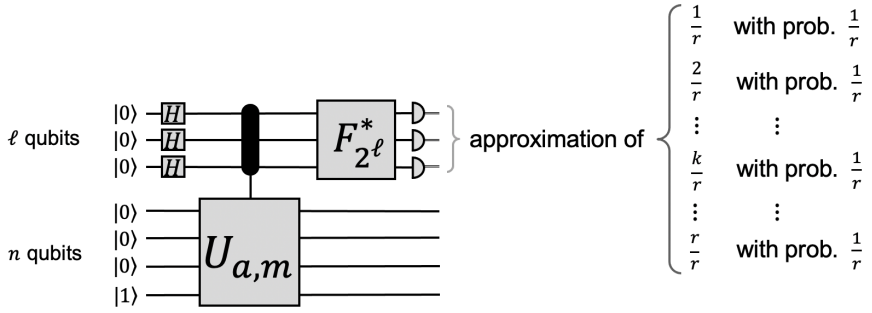

Figure 178: Caption.

The accuracy parameter ℓ is set to $2n+1$. The output of the measurement is then post-processed by the classical continued fractions algorithm, which produces a numerator k and denominator r. After two runs, taking the least common multiple of the two denominators, we obtain r, with constant probability. This constant probability is based on the phase estimation algorithm succeeding, in addition to r occurring via the continued fractions algorithm. The total gate cost is $O(n^2 \log n)$.

24.4.1 A technicality about the multiplicity-controlled-$U_{a,m}$ gate

In section 24.1.1, I glossed over some technical details about how the multiplicity-controlled-$U_{a,m}$ gate can be implemented. That gate is a unitary operation such that

$$|x, b\rangle \mapsto |x, a^x b\rangle, \tag{305}$$

for all $x \in \{0,1\}^\ell$ and $b \in \mathbb{Z}_m^*$. But our framework for computing classical functions in superposition (in sections 21.5.2 and 21.5.3) is different from this. What we showed there is that we can compute an f-query

$$|x, c\rangle \mapsto |x\rangle\, |f(x) \oplus c\rangle \tag{306}$$

in terms of a classical implementation of f.

There are two remedies. One is a separate construction for efficiently computing the mapping in Eq. (305). Such a efficient construction exists; however, I will not explain it here. Instead, I will show how to use a mapping of the form in Eq. (306) *instead of* the mapping in Eq. (305).

Define $f_{a,m} : \{0,1\}^{\ell} \to \{0,1\}^{n}$ as $f_{a,m}(x) = a^x$. Then, since there is a classical algorithm for efficiently computing $a^x b \bmod m$ in terms of (x, a, b) we can efficiently compute the mapping $|x, c\rangle \mapsto |x, f_{a,m}(x) \oplus c\rangle$. And then we can use the circuit in figure 179 instead of the circuit in figure 178.

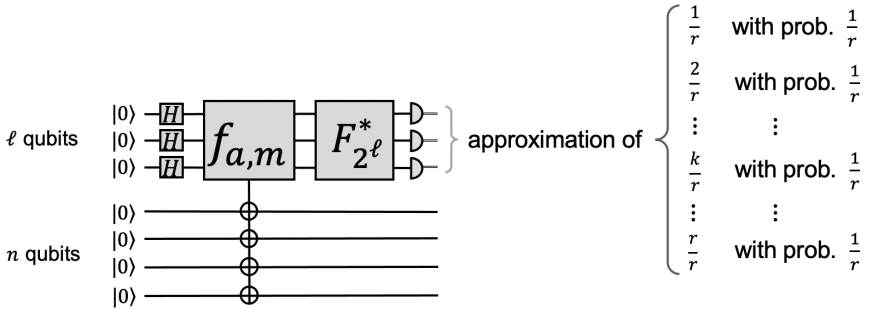

Figure 179: Caption.

The outputs of the two circuits are the same because, for both circuits, the state right after the multiplicity-controlled-$U_{a,m}$ gate (in figure 178) and the state right after the $f_{a,m}$-query (in figure 179) are the same, namely

$$\frac{1}{\sqrt{2^\ell}} \sum_{x \in \{0,1\}^\ell} |x\rangle \, |a^x\rangle \, . \tag{307}$$

25 Shor's factoring algorithm

Figure 180 describes Shor's factoring algorithm[32] (for factoring an n-bit number m), where the heavy lifting is done by a quantum subroutine for the order-finding problem (section 24). The algorithm is based on the reduction of the factoring problem to the order-finding problem, which is explained in section 23.2.

choose a random $a \in \{1, 2, \ldots, m-1\}$
$g \leftarrow \gcd(a, m)$
if $g = 1$ **then** (case where $a \in \mathbb{Z}_m^*$)
    $r \leftarrow \mathrm{ord}_m(a)$ (the quantum part of the algorithm)
    if r is even **then**
        $f \leftarrow \gcd(a^{r/2} + 1, m)$
        if f properly divides m **then**
            output f
        end if
    end if
else (case where $g > 1$, so g is a proper divisor of m)
    output g
end if

Figure 180: Factoring algorithm (using quantum algorithm for order-finding as a subroutine).

The algorithm samples uniformly from $\mathbb{Z}_m^*$ by sampling an a uniformly from the simpler set $\{1, 2, \ldots, m-1\}$ and then checking whether or not $\gcd(a, m) = 1$. If $\gcd(a, m) = 1$ then $a \in \mathbb{Z}_m^*$ and the algorithm proceeds. If $\gcd(a, m) > 1$ then, although the algorithm could randomly sample another element from $\{1, 2, \ldots, m-1\}$, that's not necessary because, in that event, $\gcd(a, m)$ is a proper divisor of m.

In the event that $a \in \mathbb{Z}_m^*$, the probability that a is lucky (as defined in section 23.2) is at least $\frac{1}{2}$. If a is lucky then the algorithm computes $\gcd(a^{r/2} + 1, m)$ to obtain a factor of m. This part can be computed by repeated squaring and Euclid's algorithm.

The implementation cost of this algorithm is $O(n^2 \log n)$ gates. Although we only analysed this algorithm for the case where m is the product of two distinct primes, it turns out that this factoring algorithm succeeds with probability at least $\frac{1}{2}$ for *any* number m that's not a prime power. For the prime power case (where $m = p^k$ for a prime p) there's a simple classical algorithm. That algorithm can be run as a preprocessing step to the algorithm in figure 180.

26 Grover's search algorithm

In this section, I will show you Grover's search algorithm,[38] which solves several search problems quadratically faster than classical algorithms. Although quadratic improvement is less dramatic than the exponential improvement possible by Shor's algorithms for factoring and discrete log, Grover's algorithm is applicable to a very large number of problems.

Let's begin by defining an abstract black-box search problem. You are given a black-box computing a function $f : \{0,1\}^n \to \{0,1\}$. Here's what the black-box looks like in the classical case (in reversible form) and the quantum case.

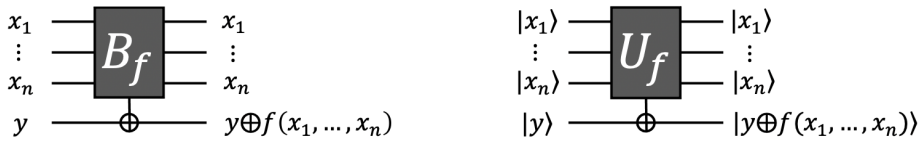

Figure 181: Classical reversible f-query (left) and quantum f-query (right).

In the notation of figure 181, $B_f : \{0,1\}^{n+1} \to \{0,1\}^{n+1}$ is a bijection and U_f is a unitary operation acting on $n+1$ qubits.

The goal is to find an $x \in \{0,1\}^n$ such that $f(x) = 1$. This is called a *satisfying assignment* (or *satisfying input*) because it is a setting of the logical variables so that the function value is "true" (denoted by the bit 1). It is possible for the function to be the constant zero function, in which case there does not exist a satisfying assignment. In that case, the solution to the problem is to report that f is "unsatisfiable". There is also a related *decision problem*, where the goal is to simply determine whether or not f is satisfiable. For that problem, the answer is one bit.

How many classical queries are needed to solve this search problem? It turns out the 2^n queries are necessary in the worst case. If f is queried in $2^n - 1$ places and the output is 0 for each of these queries then there's no way of knowing whether f is satisfiable or not without querying f in the last place.

Remember back in section 17.4 we saw that there can be a dramatic difference between the classical deterministic query cost and the classical probabilistic cost? For the constant-vs-balanced problem, we saw that exponentially many queries are necessary for for an exact solution, but that there is a classical probabilistic algorithm that succeeds with probability (say) $\frac{3}{4}$ using only a constant number of queries.

[38] L. K. Grover (1996). "A fast quantum mechanical algorithm for database search." *Proc. 28th Ann. ACM Symp. on Theory of Computing*, pp. 212–219. arXiv:9605043

So how does the probabilistic query cost work out for this search problem? What's the classical probabilistic query cost if we need only succeed with probability $\frac{3}{4}$? It turns out that order 2^n queries are still needed. Suppose that f is satisfiable in exactly one place that was set at random. Then, by querying f in random places, on average, the satisfying assignment will be found after searching half of the inputs. So the *expected number* of queries is around $\frac{1}{2}2^n$. Based on these ideas, it can be proven that a constant times 2^n queries are *necessary* to find a satisfying x with success probability at least $\frac{3}{4}$. So, asymptotically, this problem is just as hard for probabilistic algorithms as for deterministic algorithms.

What's the quantum query cost? We will see that it's $O(\sqrt{2^n})$ by Grover's algorithm, which is the subject of the rest of this section. Notice that the query cost is still exponential. The difference is only by a factor of 2 in the exponent. But this speed-up should not be dismissed. If you're familiar with algorithms then you are probably aware of clever fast sorting algorithms that make $O(n \log n)$ steps instead of the $O(n^2)$ steps that you get if you use the most obvious algorithm. The improvement is only quadratic, but for large n this quadratic improvement can make a huge difference. Similarly, in signal processing, there's a famous *Fast Fourier Transform algorithm*, whose speed-up is again quadratic over trivial approaches to the same problem.

A natural application of Grover's algorithm is when a function $f : \{0,1\}^n \to \{0,1\}$ can be computed in polynomial-time, so there is a quantum circuit of size $O(n^c)$ that implements an f-query.

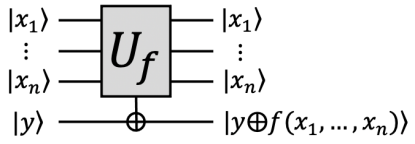

Figure 182: Implementation of a quantum f-query by a quantum circuit.

We can use Grover's algorithm to find a satisfying assignment with a circuit of size $O(\sqrt{2^n}n^c)$ (which is $O(\sqrt{N}\log^c N)$ if $N = 2^n$). For many of the so-called NP-complete problems, this is quadratically faster than the best classical algorithm known.

The order-finding algorithm and factoring algorithm use interesting properties of the Fourier transform. Grover's algorithm uses properties of simpler transformations.

26.1 Two reflections is a rotation

Consider the two-dimensional plane. There's a transformation called a *reflection about a line*. The way it works is: every point on one side of the line is mapped to a mirror image point on the other side of the line, as illustrated in figure 183.

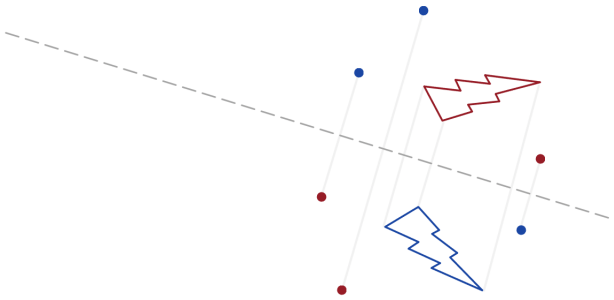

Figure 183: The reflection about the line of each red item is shown in blue.

Now suppose that we add a second line of reflection, with angle θ between the lines.

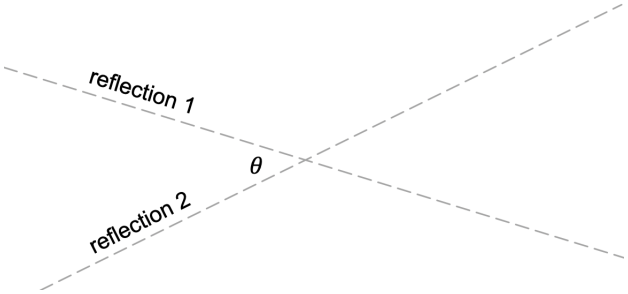

Figure 184: Two refection lines with angle θ between them.

What happens if you compose the two reflections? That is, you apply reflection 1 and then reflection 2? Let the *origin* be where the two lines intersect and think of each point as a vector to that point. Now, start with any such vector.

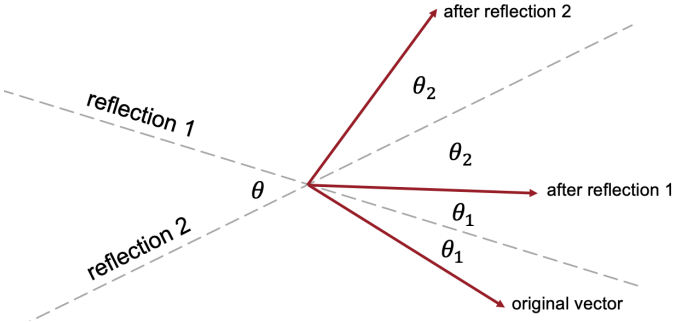

Figure 185: Effect of reflection 1 followed by reflection 2 on a vector.

This vector makes some angle—call it θ_1—with the first line. After we perform the first reflection, the reflected vector makes the same angle θ_1 on the other side of the line. The reflected vector also makes some angle with the second line. Call that angle θ_2. Notice that $\theta_1 + \theta_2 = \theta$. Now, if we apply the second reflection then the twice reflected vector makes an angle θ_2 on the other side of the second line. So what happened to the original vector as a result of these two reflections? It has been rotated by angle 2θ.

Notice that the amount of the rotation depends only on θ, it does not on depend on what θ_1 and θ_2 are. This is an illustration of the following lemma.

Lemma 26.1. *For any two reflections with angle θ between them, their composition is a rotation by angle 2θ.*

The picture in Figure 185 is intuitive, but is not a rigorous proof. A rigorous proof can be obtained by expressing each reflection operation as a 2×2 matrix, and then multiplying the two matrices. The result will be a rotation matrix by angle 2θ. The details of this are left as an exercise.

Exercise 26.1. Prove Lemma 26.1.

This simple geometric result will be a key part of Grover's algorithm.

26.2 Overall structure of Grover's algorithm

Grover's algorithm is based on repeated applications of three basic operations.

One operation is the f-query that we're given as a black-box.

Another operation that will be used, is one that we'll call U_0 that is defined as, for all $a \in \{0,1\}^n$ and $b \in \{0,1\}$,

$$U_0 \left|a\right\rangle \left|b\right\rangle = \left|a\right\rangle \left|b \oplus [a = 0^n]\right\rangle, \tag{308}$$

where $[a = 0^n]$ denotes the predicate

$$[a = 0^n] = \begin{cases} 1 & \text{if } a = 0^n \\ 0 & \text{if } a \neq 0^n. \end{cases} \tag{309}$$

In other words, in the computational basis, U_0 flips the target qubit if and only if the first n qubits are all $|0\rangle$. Our notation for the U_0 gate is shown in Figure 186 (which also shows one implementation in terms of an n-ary Toffoli gate).

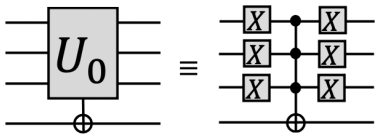

Figure 186: Notation for U_0 gate (implementable by Pauli X gates and an n-fold Toffoli gate).

A third operation that we'll use is an n-qubit Hadamard gate, by which we mean $H^{\otimes n}$ (the n-fold tensor product of 1-qubit Hadamard gates). However, in this context, we will denote this gate as H (without the superscript $\otimes n$).

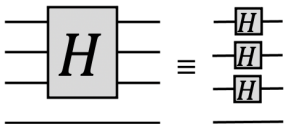

Figure 187: Notation for $H^{\otimes n}$ gate.

Also, we will use the U_f and U_0 gates with the target qubit set state $|-\rangle$, which causes the function value to be computed in the phase as, for all $x \in \{0,1\}^n$,

$$U_f |x\rangle |-\rangle = (-1)^{f(x)} |x\rangle |-\rangle \tag{310}$$

$$U_0 |x\rangle |-\rangle = (-1)^{[x=00\ldots0]} |x\rangle |-\rangle . \tag{311}$$

Here's what the main part of Grover's algorithm looks like in terms of the three basic operations.

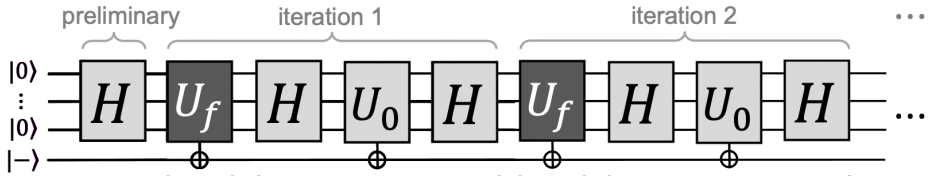

Figure 188: Quantum circuit for Grover's algorithm.

It starts with a preliminary step, which is to apply H to the state $|0...0\rangle$.

Next, the sequence of operations U_f, H, U_0, H is applied several times. Each instance of this sequence, HU_0HU_f, is called an *iteration*. After the iterations, the state of the first n qubits is measured (in the computational basis) to yield an $x \in \{0,1\}^n$. Then a classical query on input x is performed to check if $f(x) = 1$. If it is then the algorithm has solved the search problem.

Here's a summary of the algorithm in pseudocode, where k is the number of iterations performed.

1: construct state $H|00\ldots0\rangle|-\rangle$
2: **repeat** k **times:**
3: apply $-HU_0HU_f$ to the state
4: measure state, to get $x \in \{0,1\}^n$, and check if $f(x) = 1$

Figure 189: Pseudocode description of Grover's algorithm (including classical post-processing).

You might notice that, in the pseudocode, a minus sign in the iteration $-HU_0HU_f$. This is just a global phase, which has no affect on the true quantum state, but it makes the upcoming geometric picture of the algorithm nicer.

What is k, the number of iterations, set to? This is an important issue that will be addressed later on.

What I will show next is that U_f is a reflection and $-HU_0HU_f$ is also a reflection (and then we'll apply Lemma 26.1 about two reflections being a rotation). The function f partitions $\{0,1\}^n$ into two subsets, A_0 and A_1, defined as

$$A_1 = \{x \in \{0,1\}^n : f(x) = 1\} \tag{312}$$

$$A_0 = \{x \in \{0,1\}^n : f(x) = 0\}. \tag{313}$$

The set A_1 consists of the satisfying inputs to f and A_0 consists of the unsatisfying inputs to f. Let $s_1 = |A_1|$ and $s_0 = |A_0|$. Then $s_1 + s_0 = N$ (where $N = 2^n$).

Note that, if f has many satisfying assignments (for example if half the inputs are satisfying) then it's easy to find one with high probability by just random guessing. The interesting case is where there are very few satisfying assignments to f. To understand Grover's algorithm, it's useful to focus our attention on the case where s_1 is at least 1 but much smaller than N (for example, if $s_1 = 1$ or a constant). In that case, finding a satisfying assignment is nontrivial.

Now let's define these two quantum states

$$|A_1\rangle = \tfrac{1}{\sqrt{s_1}} \sum_{x \in A_1} |x\rangle \tag{314}$$

$$|A_0\rangle = \tfrac{1}{\sqrt{s_0}} \sum_{x \in A_0} |x\rangle\,. \tag{315}$$

The states $|A_1\rangle$ and $|A_0\rangle$ make sense as orthonormal vectors as long as $0 < s_1 < N$.

Consider the two-dimensional space spanned by $|A_1\rangle$ and $|A_0\rangle$. Notice that $H|0...0\rangle$ (the state of the first n qubits right after the preliminary Hadamard operations) resides within this two-dimensional space, since

$$\frac{1}{\sqrt{N}} \sum_{x\in\{0,1\}^n} |x\rangle = \sqrt{\frac{s_0}{N}}\left(\frac{1}{\sqrt{s_0}} \sum_{x\in A_0} |x\rangle\right) + \sqrt{\frac{s_1}{N}}\left(\frac{1}{\sqrt{s_1}} \sum_{x\in A_1} |x\rangle\right) \tag{316}$$

$$= \sqrt{\frac{s_0}{N}}\,|A_0\rangle + \sqrt{\frac{s_1}{N}}\,|A_1\rangle\,. \tag{317}$$

Let θ be the angle between $|A_0\rangle$ and $H|0...0\rangle$, as illustrated in figure 190.

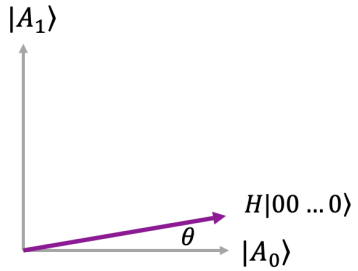

Figure 190: The state $H|0...0\rangle$ within the subspace spanned by $|A_0\rangle$ and $|A_1\rangle$.

Then $\cos(\theta) = \sqrt{\frac{s_0}{N}}$ and $\sin(\theta) = \sqrt{\frac{s_1}{N}}$. In the case where $1 \leq s_1 \ll N$, the angle θ is small and approximately $\sqrt{\frac{s_1}{N}}$.

If the system were in state $|A_0\rangle$ then measuring (in the computational basis) would result in a random unsatisfying assignment. If the system were in state $|A_1\rangle$ then measuring would result in a random satisfying assignment. After the preliminary step, the system is actually in state $H|0...0\rangle$, and measuring at that point would result in an unsatisfying assignment with high probability.

A useful goal is to somehow rotate the state $H|0...0\rangle$ towards $|A_1\rangle$. Note that, although the states $|A_0\rangle$ and $|A_1\rangle$ exist somewhere within the 2^n-dimensional space of all states of the first n qubits, the algorithm does not have information about what they are. It's not possible to directly apply a rotation matrix in the two-dimensional space without knowing what $|A_0\rangle$ and $|A_1\rangle$ are.

The key idea in Grover's algorithm is that the iterative step $-HU_0HU_f$ consists of two reflections in this two-dimensional space, whose composition results in a rotation. In the analysis below, we omit the last qubit, whose state is $|-\rangle$ and doesn't change. We write $U_f|x\rangle = (-1)^{f(x)}|x\rangle$ as an abbreviation of $U_f|x\rangle|-\rangle = (-1)^{f(x)}|x\rangle|-\rangle$ (and similarly for U_0).

The first reflection is U_f, which is a reflection about the line in the direction of $|A_0\rangle$. Why? Because $U_f |A_0\rangle = |A_0\rangle$ and $U_f |A_1\rangle = -|A_1\rangle$.

The second reflection is $-HU_0H$. This is a reflection about the line in the direction of $H|0...0\rangle$. To see this requires a little bit more analysis.

Lemma 26.2. *$-HU_0H$ is a reflection about the line passing through $H|0...0\rangle$.*

Proof. First, note that $(-HU_0H)H|0...0\rangle = H|0...0\rangle$ because

$$(-HU_0H)H|0...0\rangle = -HU_0|0...0\rangle \tag{318}$$
$$= -H(-|0...0\rangle) \tag{319}$$
$$= H|0...0\rangle. \tag{320}$$

Next, consider any vector v that is orthogonal to $H|0...0\rangle$. Then Hv is orthogonal to $|0...0\rangle$. Therefore, $U_0Hv = Hv$, from which it follows that $-HU_0Hv = -v$. We have shown that the effect of $-HU_0H$ on any vector w is to leave the component of w in the direction of $H|0...0\rangle$ fixed and to multiply any orthogonal component by -1. ■

Now, since each iteration $-HU_0HU_f$ is a composition of two reflections, and the angle between them is θ, the effect of $-HU_0HU_f$ is a rotation by 2θ. And the effect of k iterations is to rotate by $k2\theta$, as illustrated in figure 191.

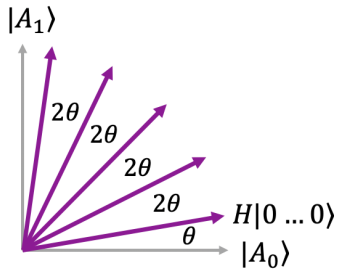

Figure 191: The state $|0...0\rangle$ rotates by angle 2θ after each iteration.

What should k be set to? How many iterations should the algorithm make to move the state close to $|A_1\rangle$? Let us begin by focusing on the case where there is one unique satisfying input to f.

26.3 The case of a unique satisfying input

Consider the case where there is one satisfying assignment (that is, where $s_1 = 1$). In that case, $\sin(\theta) = \frac{1}{\sqrt{N}}$. Since $\frac{1}{\sqrt{N}}$ is small, we have $\theta \approx \frac{1}{\sqrt{N}}$.

Then setting $k = \frac{\pi}{4}\sqrt{N}$ (rounded to the nearest integer) causes the total rotation to be approximately $\frac{\pi}{2}$ radians (that is, 90 degrees). The resulting state will not be exactly $|A_1\rangle$. In most cases, $|A_1\rangle$ will be somewhere between two rotation angles. But the state will be close to $|A_1\rangle$, so measuring will produce a satisfying assignment with high probability (certainly at least $\frac{3}{4}$).

The resulting algorithm finds the satisfying x with high probability and makes $O(\sqrt{N})$ queries to f (one per iteration).

26.4 The case of any number of satisfying inputs

What if f has more than one satisfying input? One might be tempted to think intuitively that the "hardest" case for the search is where there is just one satisfying input. Finding a needle in a haystack is harder if the haystack contains a single needle. So how well does the algorithm with $k \approx \frac{\pi}{4}\sqrt{N}$ iterations do when there are more satisfying inputs? Unfortunately, it's not very good.

Suppose that there are four satisfying inputs. Then $\theta \approx \frac{2}{\sqrt{N}}$ (double the value in the case of one satisfying input), which causes the rotation to overshoot. The total rotation is by 180 degrees instead of 90 degrees. The state starts off almost orthogonal to $|A_1\rangle$. Then, midway, it gets close to $|A_1\rangle$. And then it keeps on rotating and becomes almost orthogonal to $|A_1\rangle$ again.

When there are four satisfying inputs, the ideal number of iterations is half the number for the case of one satisfying input. More generally, when there are s satisfying inputs, the ideal number of iterations is $k \approx \frac{\pi}{4}\sqrt{N/s}$. If we know the number of satisfying inputs in advance then we can tune k appropriately.

But suppose we're given a black box for f without any information about the number of satisfying inputs that f. What do we do then? For any particular setting of k, it's concievable that number of satisfying inputs to f is such that the total rotation after k iterations is close to a multiple of 180 degrees.

The case of an unknown number of satisfying inputs has been addressed in detail.[39] I will roughly sketch a simplified approach that attains $O(\sqrt{N})$ iterations by setting k to a random element of the set $\left\{1, 2, \ldots, \lceil \frac{\pi}{4}\sqrt{N} \rceil\right\}$.

To see how this works, let's begin by considering again the case where there is one satisfying input. Figure 192 shows the success probability as a function of the

[39] M. Boyer, G. Brassard, P. Høyer, and A. Tapp (1996). "Tight bounds on quantum searching." *Proc. 4th Workshop on Physics and Computation*, pp. 36–43. arXiv:9605034

number iterations k, for any $k \in \left\{1, 2, \ldots, \lceil \frac{\pi}{4}\sqrt{N} \rceil\right\}$.

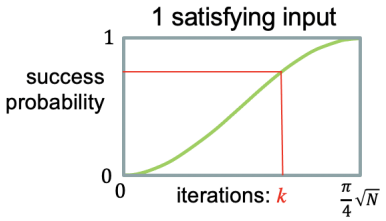

Figure 192: Success probability of measurement as a function of k, the number of iterations.

The curve is sinusoidal (of the form of a sine function). In the case of a single satisfying input, setting $k = \lceil \frac{\pi}{4}\sqrt{N} \rceil$ results in a success probability close to 1. But if k is set randomly then the success probability is the average value of the curve. By the symmetry of the sine curve, the area under the curve is half of the area of the box, so the success probability is $\frac{1}{2}$.

Now, let's consider the cases of two, three or four satisfying assignments. These cases result in a larger θ, and the corresponding success probability curves are shown in figure 193 (they are sinusoidal curves corresponding to more total rotation).

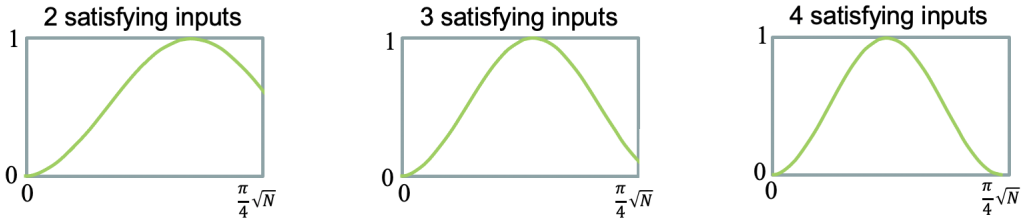

Figure 193: Success probability function in cases of 2, 3, and 4 satisfying inputs.

By inspection, the area under each of these curves is at least $\frac{1}{2}$. Each curve is a 90 degree rotation, for which the average is $\frac{1}{2}$, followed by another rotation less than or equal to 90 degrees, for which the average can be seen to be more than $\frac{1}{2}$.

But there are cases where the average is less than $\frac{1}{2}$. Let's look at the case of six satisfying inputs.

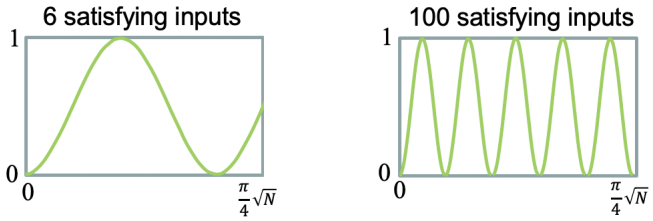

Figure 194: Success probability function in cases of 6 and 100 satisfying inputs.

There are two 90-degree rotations, followed by a rotation less than 90 degrees for which the average is less than $\frac{1}{2}$. The average for the whole curve can be calculated to be around 0.43. For larger numbers of satisfying inputs, the proportion of the domain associated with the last partial rotation becomes small enough so as to be inconsequential.

The above explains at a heurustic level why choosing k randomly within the range results in a success probability that's at least 0.43, regardless of the number of satisfying assignments. That's one way to solve the case of an unknown number of satisfying inputs to f.

To summarize, we have a quantum algorithm that makes $O(\sqrt{N})$ queries and finds a satisfying assignment with constant probability. By repeating the process a constant number of times, the success probability can be made close to 1.

What happens if there is no satisfying assignment? In that case, the algorithm will not find a satisfying assignment in any run and outputs "unsatisfiable."

A more refined version of the search algorithm[39] stops early, depending on the number of satisfying inputs. The number of queries is $O\left(\sqrt{N/s}\right)$, where s is the number of satisfying inputs. This refined algorithm does not have information about the value of s. When the algorithm is run, we don't know in advance for how long it will need to run. I will not get into the details of this refined algorithm here.

Finally, you might wonder if Grover's search algorithm can be improved. Could there be a better quantum algorithm that performs the black-box search with fewer than order $\sqrt{N}$ queries? Say, $O(N^{1/3})$ or $O(\log N)$ queries? Actually, no. It can be proven that the above query costs are the best possible, as explained in the next section.

27 Optimality of Grover's search algorithm

In this section, I will show you a proof that Grover's search algorithm is optimal in terms of the number of black-box queries that it makes.[40]

Theorem 27.1 (optimality of Grover's algorithm)**.** *Any quantum algorithm for the search problem for functions of the form* $f : \{0,1\}^n \to \{0,1\}$ *that succeeds with probability at least* $\frac{3}{4}$ *must make* $\Omega\left(\sqrt{2^n}\right)$ *queries.*

In this proof, I make two simplifying assumptions. First, I assume that the function is always queried in the phase (as occurs with Grover's algorithm).

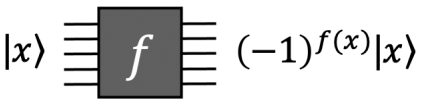

Figure 195: Query of f in the phase.

Second, I assume that the algorithm uses only n qubits. That is, no ancilla qubits are used. The purpose of these simplifying assumptions is to reduce clutter. It is very straightforward to adjust this proof to the general case, where the function is queried in the standard way, and where the quantum algorithm is allowed to use ancilla qubits. The more general proof will be just a more cluttered-up version of the proof that I'm about to explain. Nothing of importance is being swept under the rug in this simplified proof.

Under our assumptions, any quantum algorithm that makes k queries is a circuit of the following form.

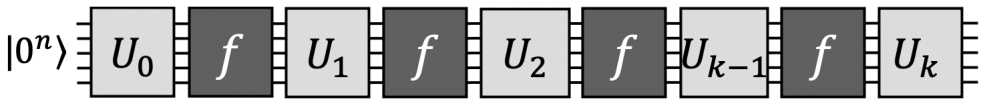

Figure 196: Form of a k query quantum circuit.

The initial state of the n qubits is $|0^n\rangle$. Then an arbitrary unitary operation U_0 is applied, which can create any pure state as the input to the first query. Then the first f-query is performed. Then an arbitrary unitary operation U_1 is applied. Then the

[40] C. H. Bennett, E. Bernstein, G. Brassard, and U. Vazirani (1993). "Strengths and weaknesses of quantum computing." *Proc. 25th Ann. ACM Symp. on Theory of Computing*, pp. 11–20. arXiv:9701001

second f-query is performed. And so on, up the k-th f-query, followed by an arbitrary unitary operation U_k. The *success probability* is the probability that a measurement of the final state results in a satisfying input of f.

The quantum algorithm is the specification of the unitaries $U_0, U_1, \ldots, U_k$. The *input* to the algorithm is the black-box for f, which is inserted into the k query gates. The *quantum output* is the final state produced by the quantum circuit.

The overall approach will be as follows. For every $r \in \{0,1\}^n$, define f_r as

$$f_r(x) = \begin{cases} 1 & \text{if } x = r \\ 0 & \text{otherwise.} \end{cases} \tag{321}$$

We'll show that, for any algorithm that makes asymptotically fewer than $\sqrt{2^n}$ queries, it's success probability is very low for at least one f_r.

Assume we have some k-query algorithm, specified by $U_0, U_1, \ldots, U_k$. Suppose that we insert one of the functions f_r into the query gate.

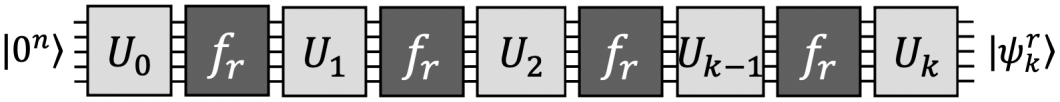

Figure 197: Quantum circuit making k queries to function f_r.

Call the final state $|\psi_k^r\rangle$. The superscript r is because this state depends on which r is selected. The subscript k is to emphasize that k queries are made.

Now consider the exact same algorithm, run with the identity operation in each query gate.

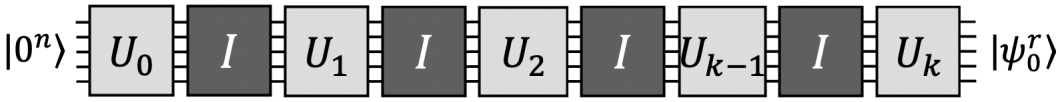

Figure 198: Quantum circuit with I substituted for the k queries.

Whenever the identity is substituted into a query gate, we'll call that a *blank query*. Let's call the final state produced by this $|\psi_0^r\rangle$. Although the superscript r is redundant for this state (because the state is not a function of r), it's harmless and it will be convenient to keep this superscript r in our notation. The circuit in figure 197 corresponds to the function f_r. The circuit in figure 198 corresponds to the zero function (the function that's zero everywhere).

What we are going to show is that, for some $r \in \{0,1\}^n$, the Euclidean distance between $|\psi_k^r\rangle$ and $|\psi_0^r\rangle$ is upper bounded as

$$\left\| |\psi_k^r\rangle - |\psi_0^r\rangle \right\| \leq \frac{2k}{\sqrt{2^n}}. \tag{322}$$

This implies that, if k is asymptotically smaller than $\sqrt{2^n}$ then the bound asymptotically approaches zero. If $\| |\psi_k^r\rangle - |\psi_0^r\rangle \|$ approaches zero then the two states are not distinguishable in the limit. This implies that the algorithm cannot distinguish between f_r and the zero function with constant probability.

Now let's consider the quantum circuits in figures 197 and 198, along with some intermediate circuits.

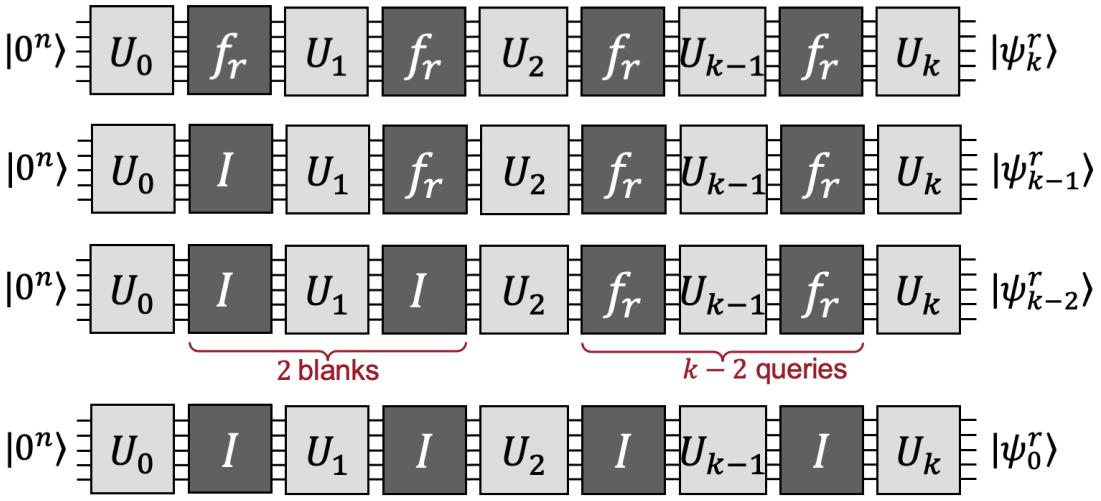

Figure 199: Quantum circuits with i blanks, followed by $k-i$ queries to f_r (for $i \in \{0,1,\ldots,r\}$).

The circuit on top makes all k queries to f_r, and recall that we denote the final state that it produces as $|\psi_k^r\rangle$. The next circuit uses the identity in place of the *first* query and makes the remaining $k-1$ queries to f_r. Call the final state of this $|\psi_{k-1}^r\rangle$. The next circuit uses the identity in place of the first *two* queries and makes the remaining $k-2$ queries to f_r. Call the final state that this produces $|\psi_{k-2}^r\rangle$. And continue for $i = 3,\ldots,k$ with circuits using the identity in place of the first i queries and then making the remaining $k-i$ queries to f_r. Call the final states $|\psi_{k-i}^r\rangle$.

Note that each query where the identity is substituted (blanks query) reveals no information about r.

Recall that our goal is to show that the $\| \, |\psi_k^r\rangle - |\psi_0^r\rangle \, \|$ is small. Notice that we can express the difference between these states as the telescoping sum

$$|\psi_k^r\rangle - |\psi_0^r\rangle = \Big(|\psi_k^r\rangle - |\psi_{k-1}^r\rangle\Big) + \Big(|\psi_{k-1}^r\rangle - |\psi_{k-2}^r\rangle\Big) + \cdots + \Big(|\psi_1^r\rangle - |\psi_0^r\rangle\Big) \tag{323}$$

which implies that

$$\Big\||\psi_k^r\rangle - |\psi_0^r\rangle\Big\| \leq \Big\||\psi_k^r\rangle - |\psi_{k-1}^r\rangle\Big\| + \Big\||\psi_{k-1}^r\rangle - |\psi_{k-2}^r\rangle\Big\| + \cdots + \Big\||\psi_1^r\rangle - |\psi_0^r\rangle\Big\|. \tag{324}$$

Next, we'll upper bound each term $\| \, |\psi_i^r\rangle - |\psi_{i-1}^r\rangle \|$ in Eq. 324. To understand the Euclidean distance between $|\psi_i^r\rangle$ and $|\psi_{i-1}^r\rangle$, let's look at the circuits that produce them.

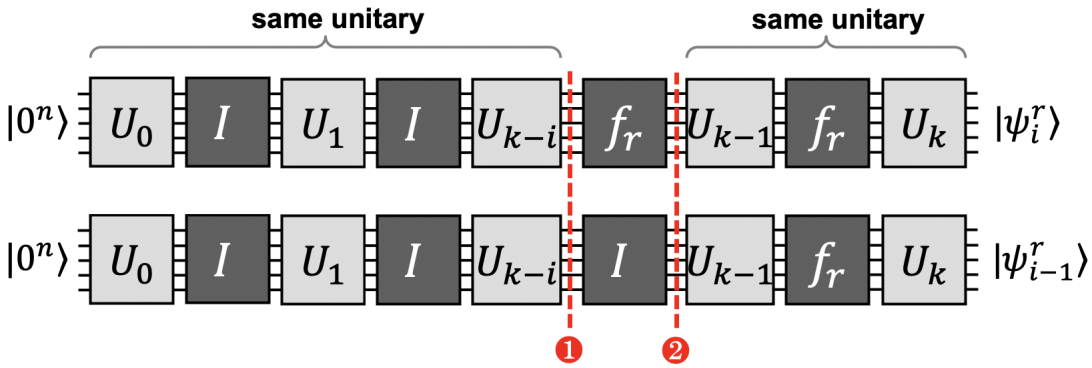

Figure 200: Circuit with $k-i$ blanks and i queries (top) and $k-i+1$ blanks and $i-1$ queries (bottom).

Notice that both computations are identical for the first $k-i$ blank queries. We can write the state at stage ❶ (for both circuits) as

$$\sum_{x \in \{0,1\}^n} \alpha_{i,x} \, |x\rangle \, , \tag{325}$$

where of course the state is a unit vector in Euclidean norm (a.k.a, the 2-norm)

$$\sum_{x \in \{0,1\}^n} |\alpha_{i,x}|^2 = 1. \tag{326}$$

These states make sense for each $i \in \{1, \ldots, k\}$, and they are independent of r.

It's interesting to consider what happens to the state at the next step. For the bottom computation, nothing happens to the state, since the identity is performed in the query. For the top computation, a query to f_r is performed. Since f_r takes value 1 only at the point r, the f_r-query in the phase negates the amplitude of $|r\rangle$

and has no effect on the other amplitudes of the state. Therefore, the state of the top computation at stage ❷ is

$$\sum_{x\in\{0,1\}^n} (-1)^{[x=r]}\alpha_{i,x}\,|x\rangle\,. \tag{327}$$

The difference between the state in Eq. (325) and the state in Eq. (327) is only in the amplitude of $|r\rangle$, which is negated in Eq. (327). So, at stage ❷ the Euclidean distance between the states of the two circuits is $|\alpha_{r,i} - (-\alpha_{r,i})| = 2|\alpha_{r,i}|$.

Now let's look at what the rest of the two circuits in figure 200 do. The remaining steps for each circuit are the same unitary operation. Since unitary operations preserve Euclidean distances, this means that

$$\left\||\psi_i^r\rangle - |\psi_{i-1}^r\rangle\right\| = 2|\alpha_{i,r}|. \tag{328}$$

And this holds for all $i \in \{1,\ldots,k\}$.

Now consider the average Euclidean distance between $|\psi_k^r\rangle$ and $|\psi_0^r\rangle$, averaged over all n-bit strings r. We can upper bound this as

$$\frac{1}{2^n}\sum_{r\in\{0,1\}^n}\left\||\psi_k^r\rangle - |\psi_0^r\rangle\right\| \leq \frac{1}{2^n}\sum_{r\in\{0,1\}^n}\left(\sum_{i=1}^{k}\left\||\psi_i^r\rangle - |\psi_{i-1}^r\rangle\right\|\right) \quad \text{telescoping sum} \tag{329}$$

$$= \frac{1}{2^n}\sum_{r\in\{0,1\}^n}\left(\sum_{i=1}^{k} 2|\alpha_{i,r}|\right) \quad \text{by Eq. (328)} \tag{330}$$

$$= \frac{1}{2^n}\sum_{i=1}^{k} 2\left(\sum_{r\in\{0,1\}^n}|\alpha_{i,r}|\right) \quad \text{reordering sums} \tag{331}$$

$$\leq \frac{1}{2^n}\sum_{i=1}^{k} 2\left(\sqrt{2^n}\right) \quad \text{Cauchy-Schwarz} \tag{332}$$

$$= \frac{2k}{\sqrt{2^n}}. \tag{333}$$

In Eq. (332) we are using the the Cauchy-Schwarz inequality, which implies that, for any vector whose 2-norm is 1, its maximum possible 1-norm is the square root of the dimension of the space, which in this case is $\sqrt{2^n}$.

Since an average of Euclidean distances is upper bounded by $2k/\sqrt{2^n}$, there must exist an r for which the Euclidean distance upper bounded by this amount.

Since the denominator is $\sqrt{2^n}$ and the numerator is $2k$, it must be the case that k is proportional to $\sqrt{2^n}$, in order for this Euclidean distance to be a constant. And

the Euclidean must be lower bounded by a constant if the algorithm distinguishes between f_r and the zero function with probability $\frac{3}{4}$. This completes the proof that the number of queries k much be order $\sqrt{2^n}$.

Part III
Quantum information theory

28 Quantum states as density matrices

Let's begin by considering a couple of situations where Alice has an apparatus for creating quantum states for Bob, who has no apparatus. When Bob needs a specific state, he asks Alice to create it and send it to him.

Figure 201: Alice uses her apparatus to prepare states for Bob.

The story of the fake-plus state

Suppose that Bob asks Alice to create a qubit in state $|+\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and send it to him. But suppose that Alice's state preparation device is broken in that it can *only* prepare qubits in state $|0\rangle$ or $|1\rangle$.

What is Alice to do? She cannot create the state $|+\rangle$ literally. Suppose she tries to fake it by flipping a fair coin and then creating either $|0\rangle$ or $|1\rangle$, depending on the coin's outcome—while keeping the coin's outcome secret from Bob. The state that Alice creates can be described as

$$\begin{cases} |0\rangle & \text{with probability } \frac{1}{2} \\ |1\rangle & \text{with probability } \frac{1}{2}. \end{cases} \tag{334}$$

Let's call this the *fake-plus state*.

How good is this fake-plus state as a substitution for the real plus state $|+\rangle$? Is there any way that Bob can tell the difference? If, for any measurement that Bob can perform, the outcome probabilities are exactly the same then the substitution is a good one. Note that if Bob measures the fake plus state in the computational basis then the outcome probabilities are the same as measuring $|+\rangle$ in the computational basis. So far so good.

But there are other measurements for which the outcome probabilities are different. Can you think of one?

Exercise 28.1. Give a measurement which has different outcome probabilities for the fake-plus state (334) than for the true plus state $|+\rangle$.

So the fake-plus state is *not* a good substitute for the plus state.

The story of the fake-fake-plus state

Now, let's consider a different scenario. Suppose that Bob doesn't want a $|+\rangle$ state; instead, he wants Alice to prepare for him a fake-plus state, the state in Eq. (334). But this time, let's suppose that Alice's apparatus is broken in a different way: it can *only* prepare $|+\rangle$ and $|-\rangle$ states.

What can Alice do in this case? It won't do to send Bob a $|+\rangle$ state, because we already know that the plus state and the fake-plus state do not behave the same for all measurements. What if Alice tries to fake it this time by flipping a fair coin and then sending $|+\rangle$ or $|-\rangle$, depending on the outcome (again keeping the coin outcome secret). Such a state can be described as

$$\begin{cases} |+\rangle & \text{with probability } \frac{1}{2} \\ |-\rangle & \text{with probability } \frac{1}{2}. \end{cases} \tag{335}$$

Let's call this the *fake-fake-plus state*. Is the fake-fake-plus state (335) a good substitute for the fake-plus state (334)?

The two states certainly don't look the same. So your first guess might be that there's a measurement for which the outcome probabilities are different. But it turns out that, for *every* measurement that Bob can make, the outcome probabilities for state (334) are exactly the same as they are for state (335). Even though the two states don't look the same, the fake-fake-plus state is a good substitute for the fake-plus state.

28.1 Probabilistic mixtures of states

We are now in the realm of *probabilistic mixtures* of states. These are states where a random process is used to decide which pure state to prepare. Let $(p_1, p_2, ..., p_m)$ be a probability vector and let $|\psi_1\rangle, |\psi_1\rangle, \ldots, |\psi_m\rangle$ be d-dimensional quantum states (they need not be orthogonal). Imagine that $k \in \{1, 2, \ldots, m\}$ is sampled according the the probability distribution $(p_1, p_2, ..., p_m)$, and then state $|\psi_k\rangle$ is produced (but k is not revealed). Such a state can be described as

$$\begin{cases} |\psi_1\rangle & \text{with probability } p_1 \\ |\psi_2\rangle & \text{with probability } p_2 \\ \vdots & \vdots \qquad\qquad \vdots \\ |\psi_m\rangle & \text{with probability } p_m. \end{cases} \tag{336}$$

These states are called *mixed states*. The "ordinary" states, describable by a single normalized vector $|\psi\rangle$, are called *pure states*. In Eq. (336), if one of the probabilities is 1 and any others are 0 then the state is a pure state.

Let's look at some examples of probabilistic mixtures of states. We have already seen these two mixed states:

$$\begin{cases} |0\rangle & \text{with probability } \frac{1}{2} \\ |1\rangle & \text{with probability } \frac{1}{2} \end{cases} \quad \text{and} \quad \begin{cases} |+\rangle & \text{with probability } \frac{1}{2} \\ |-\rangle & \text{with probability } \frac{1}{2}, \end{cases} \tag{337}$$

and I claimed that they are indistinguishable—but I did not explain why. How do these two states compare with

$$\begin{cases} |0\rangle & \text{with probability } \frac{1}{2} \\ |-\rangle & \text{with probability } \frac{1}{2}? \end{cases} \tag{338}$$

Are they also indistinguishable from this state?

Mixed states for qubits can be probability distributions on any number of vectors, for example

$$\begin{cases} |0\rangle & \text{with prob. } \frac{1}{4} \\ |1\rangle & \text{with prob. } \frac{1}{4} \\ |+\rangle & \text{with prob. } \frac{1}{4} \\ |-\rangle & \text{with prob. } \frac{1}{4} \end{cases} \quad \text{and} \quad \begin{cases} |0\rangle & \text{with prob. } \frac{1}{3} \\ -\frac{1}{2}|0\rangle + \frac{\sqrt{3}}{2}|1\rangle & \text{with prob. } \frac{1}{3} \\ -\frac{1}{2}|0\rangle - \frac{\sqrt{3}}{2}|1\rangle & \text{with prob. } \frac{1}{3}. \end{cases} \tag{339}$$

And the probability distribution need not be uniform, as shown in this example

$$\begin{cases} \cos(\frac{\pi}{8})|0\rangle - \sin(\frac{\pi}{8})|1\rangle & \text{with prob. } \cos^2(\frac{\pi}{8}) \\ \sin(\frac{\pi}{8})|0\rangle + \cos(\frac{\pi}{8})|1\rangle & \text{with prob. } \sin^2(\frac{\pi}{8}). \end{cases} \tag{340}$$

Definition 28.1 (indistinguishable states)**.** Two probabilistic mixtures of states are *indistinguishable* if, for all possible measurements, the outcome probabilities are the same for the two states.

Now, consider the six mixed states appearing in in (337)(338)(339)(340) above. Which pairs are indistinguishable? To address such questions, we need to understand these kinds of states better. A very useful approach is to express these states in terms of their *density matrices*.

28.2 Density matrices

Given a probability distribution on a set of state vectors, one might be tempted to consider the *weighted average* of the state vectors $p_1 |\psi_1\rangle + p_2 |\psi_2\rangle + \cdots + p_m |\psi_m\rangle$ as a useful object. However, this kind of average turns out to be of little use. One indicator that it's not worth much is that the weighted average can change dramatically depending on the global phases associated with the vectors—which shouldn't matter. Also, notice that, for the second mixed state in Eq. (339), the weighted average is the zero vector.

Mixed states can be nicely characterized by a different kind of averaging, which occurs in the definition of the density matrix.

Definition 28.2 (density matrix)**.** For a mixed state of the form of Eq. (336), where $|\psi_1\rangle, |\psi_1\rangle, \ldots, |\psi_m\rangle$ are d-dimensional, its *density matrix* is the $d \times d$ matrix

$$\rho = p_1 |\psi_1\rangle \langle\psi_1| + p_2 |\psi_2\rangle \langle\psi_2| + \cdots + p_m |\psi_m\rangle \langle\psi_m| . \tag{341}$$

Note that the density matrix of any pure state $|\psi\rangle$ is $|\psi\rangle \langle\psi|$. For example, the state $|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$ has density matrix

$$\rho = \Big(\alpha_0 |0\rangle + \alpha_1 |1\rangle\Big)\Big(\bar{\alpha}_0 \langle 0| + \bar{\alpha}_1 \langle 1|\Big) \tag{342}$$

$$= \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix} \begin{bmatrix} \bar{\alpha}_0 & \bar{\alpha}_1 \end{bmatrix} \tag{343}$$

$$= \begin{bmatrix} |\alpha_0|^2 & \alpha_0 \bar{\alpha}_1 \\ \alpha_1 \bar{\alpha}_0 & |\alpha_1|^2 \end{bmatrix} . \tag{344}$$

The entries along the diagonal are the absolute values squared of the amplitudes and the off-diagonal entries are cross-terms involving the amplitudes.

Also note from Definition 28.2 that the density matrix of a probabilistic mixture of pure states is the weighted average of the density matrices of the pure states. For example, the density matrices of $|0\rangle$ and $|1\rangle$ are

$$|0\rangle \langle 0| = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \qquad \text{and} \qquad |1\rangle \langle 1| = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{345}$$

and the density matrix of the first mixed state in Eq. (337) is

$$\tfrac{1}{2} |0\rangle\langle 0| + \tfrac{1}{2} |1\rangle\langle 1| = \tfrac{1}{2} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \tfrac{1}{2} \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} . \tag{346}$$

Regarding the second mixed state in Eq. (337), the density matrices of $|+\rangle$ and $|-\rangle$ are

$$|+\rangle\langle+| = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \qquad \text{and} \qquad |-\rangle\langle-| = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{347}$$

and the density matrix of their mixture is

$$\tfrac{1}{2}|+\rangle\langle+| + \tfrac{1}{2}|-\rangle\langle-| = \tfrac{1}{2}\begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} + \tfrac{1}{2}\begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}. \tag{348}$$

Notice that the density matrices for the two mixed states in Eq. (337) are the same. This is related to the fact that the states are indistinguishable—which will be explained shortly.

Exercise 28.2 (a straightforward calculation)**.** Work out the density matrices of the mixed states appearing in (338)(339)(340).

Let me make a comment about global phases in vector states. When we represent (pure) states as vectors, there's this issue that if multiply the vector by a unit complex number (of the form $e^{i\theta}$, for $\theta \in \mathbb{R}$) then it's essentially the same state; it's indistinguishable from the original state. The density matrix of $e^{i\theta}|\psi\rangle$ is

$$e^{i\theta}|\psi\rangle\langle\psi|e^{-i\theta} = |\psi\rangle\langle\psi|. \tag{349}$$

So global phases don't even show up in density matrices, which is nice. It means that, using density matrices, we don't need to define an equivalence relation to account for global phases.

Now, let's look at some examples of density matrices for higher dimensional systems than qubits. The density matrix of $|00\rangle$ is

$$|00\rangle\langle00| = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{350}$$

and the density matrices of the other computational basis states are also diagonal matrices with a 1 in one position.

The density matrix of the Bell state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ is

$$\begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{bmatrix} \tag{351}$$

and the density matrix of the state

$$\begin{cases} |00\rangle & \text{with probability } \frac{1}{2} \\ |11\rangle & \text{with probability } \frac{1}{2} \end{cases} \tag{352}$$

is

$$\frac{1}{2}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}. \tag{353}$$

If we were adopt the terminology used for state (334), we might call state (352) a *fake-Bell state*. Notice that the difference between the density matrices of the Bell state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ and the fake-Bell state (352) is in the two off-diagonal entries of their density matrices.

28.3 Information processing in terms of density matrices

Every probabilistic mixture of states has a density matrix associated with it; however, different probabilistic mixtures can result in the same density matrix (for example the two states in Eq. (337)). Here we will see that information processing on mixed states can be understood *solely* in terms of density matrices.

Suppose that we apply operation U to a probabilistic mixture of states, where U is unitary, or more generally U is an isometry.[41] How does this affect the density matrix? If we begin with a mixed state of the form

$$\begin{cases} |\psi_1\rangle & \text{with prob. } p_1 \\ |\psi_2\rangle & \text{with prob. } p_2 \\ \vdots & \vdots & \vdots \\ |\psi_m\rangle & \text{with prob. } p_m \end{cases} \tag{354}$$

[41] Isometries are defined in Definition 9.1 of *Part I*.

then applying U changes the state to

$$\begin{cases} U\left|\psi_1\right\rangle & \text{with prob. } p_1 \\ U\left|\psi_2\right\rangle & \text{with prob. } p_2 \\ \quad\vdots & \quad\vdots \qquad\quad \vdots \\ U\left|\psi_m\right\rangle & \text{with prob. } p_m. \end{cases} \tag{355}$$

This is because, whatever $\left|\psi_k\right\rangle$ is randomly selected, it gets converted to $U\left|\psi_k\right\rangle$.

Let ρ denote the density matrix of the original state. That is,

$$\rho = \sum_{j=1}^{m} p_j \left|\psi_j\right\rangle \left\langle\psi_j\right| . \tag{356}$$

Then the density matrix after U is applied is

$$\sum_{j=1}^{m} p_j \Big(U\left|\psi_j\right\rangle\Big)\Big(U\left|\psi_j\right\rangle\Big)^* = \sum_{j=1}^{m} p_j U\left|\psi_j\right\rangle \left\langle\psi_j\right| U^* \tag{357}$$

$$= U\left(\sum_{j=1}^{m} p_j \left|\psi_j\right\rangle \left\langle\psi_j\right|\right) U^* \tag{358}$$

$$= U\rho\, U^*. \tag{359}$$

What is remarkable is that the density matrix of the modified state depends *only* on the density matrix of the original state. It does not depend on what specific probabilistic mixture is used to create the original state.

Now, let's consider the effect of a measurement of a d-dimensional mixed state, with respect to the computational basis $\left|0\right\rangle, \left|1\right\rangle, \ldots, \left|d-1\right\rangle$. Let the mixture be

$$\begin{cases} \left|\psi_1\right\rangle & \text{with prob. } p_1 \\ \left|\psi_2\right\rangle & \text{with prob. } p_2 \\ \quad\vdots & \quad\vdots \qquad\quad \vdots \\ \left|\psi_m\right\rangle & \text{with prob. } p_m. \end{cases} \tag{360}$$

There are two different ways that randomness arises in such a measurement: the randomness that was used to select one of the pure states; and, the randomness that arises in the measurement process for the selected state.

If the selected state is $|\psi_j\rangle$ then the probability of measurement outcome k is

$$|\langle k|\psi_j\rangle|^2 = \langle k|\psi_j\rangle\langle\psi_j|k\rangle = \langle k|\left(|\psi_j\rangle\langle\psi_j|\right)|k\rangle \tag{361}$$

(where we are using the fact that the expressions are all products of row matrices and column matrices and that matrix multiplication is associative).

If we average this over all possibilities of $|\psi_j\rangle$ then the probability of outcome k is

$$\sum_{j=1}^{m} p_j \langle k|\left(|\psi_j\rangle\langle\psi_j|\right)|k\rangle = \langle k|\left(\sum_{j=1}^{m} p_j |\psi_j\rangle\langle\psi_j|\right)|k\rangle \tag{362}$$

$$= \langle k|\rho|k\rangle, \tag{363}$$

where ρ is the density matrix of the original state. Also, when the measurement outcome is k, the residual state is $|k\rangle\langle k|$. Once again, the result of the operation depends *only* on the density matrix of the original state. It does not depend on what specific probabilistic mixture is used to create the original state.

In summary, we have proved the following.

Theorem 28.1. *For a d-dimensional mixed state with density matrix ρ:*

- *Applying an isometry U to the state changes it to one with density matrix $U\rho U^*$.*
- *Applying a measurement in the computational basis to the state produces classical and quantum outcomes*

$$\begin{cases} \left(0,\ |0\rangle\langle 0|\right) & \text{with prob. } \langle 0|\rho|0\rangle \\ \left(1,\ |1\rangle\langle 1|\right) & \text{with prob. } \langle 1|\rho|1\rangle \\ \quad\vdots & \quad\vdots \qquad\qquad \vdots \\ \left(d-1,\ |d-1\rangle\langle d-1|\right) & \text{with prob. } \langle d-1|\rho|d-1\rangle. \end{cases} \tag{364}$$

In both cases, the result of the operation depends only on the density matrix of the state; not on the specific probabilistic mixture that generates the state.

Any measurement of a d-dimensional system can be converted to the form of first applying an isometry $U : \mathbb{C}^d \to \mathbb{C}^c$ and then measuring with respect to the computational basis in $\mathbb{C}^c$ (possibly discarding some information about the outcome). Therefore, states with the same density matrix are indistinguishable.

In particular, the fake-plus state (334) and the fake-fake-plus state (335) are indistinguishable.

28.4 Normal matrices, positive matrices, and the trace

Prior to our use of the density matrix framework, states have been vectors and matrices have been associated with the *operations performed on states* (unitaries, isometries, as well as the projectors arising in measurements). With density matrices, states themselves are represented as matrices.

Henceforth, more types of matrices will arise as we develop our models of quantum information theory. In this section, we review some useful definitions and properties of matrices that will be used.

Definition 28.3 (normal matrix)**.** A matrix $M \in \mathbb{C}^{d\times d}$ is *normal* if $M^*M = MM^*$.

An important property of normal matrices is that they are diagonalizable in some orthonormal basis.

Theorem 28.2 (spectral theorem)**.** *A matrix $M \in \mathbb{C}^{d\times d}$ is* normal *if and only if there exists a unitary $U \in \mathbb{C}^{d\times d}$ such that*

$$M = U \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_d \end{bmatrix} U^*. \tag{365}$$

Although Definition 28.3 is the common textbook definition of *normal*, the statement of Theorem 28.2 can be taken as an alternative definition.

To help understand normal matrices, it's useful to see examples of *ab*normal matrices. Consider these two:

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \qquad\qquad \begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix}. \tag{366}$$

The first matrix is not normal because it is not even diagonalizable. The second matrix is diagonalizable but not unitarily (it has eigenvectors $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ and $\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right]$, which are not orthogonal).

For any normal matrix, by Theorem 28.2, we can imagine it to be diagonal in the coordinate system of some orthonormal basis. Note that a normal matrix M is unitary (defined as $M^*M = I$) if and only if all its eigenvalues have absolute value 1 (i.e., they are points on the unit circle in $\mathbb{C}$. And a normal matrix M is Hermitian (defined as $M = M^*$) if and only if all its eigenvalues are in $\mathbb{R}$.

All density matrices have the property that they are *positive*[42] , defined as follows.

[42]In some communities, the terminology *positive semidefinite* is used instead of *positive*.

Definition 28.4 (positive)**.** A matrix $M \in \mathbb{C}^{d\times d}$ is *positive* if and only if, for all states $|\psi\rangle \in \mathbb{C}^d$, it holds that $\langle\psi| M |\psi\rangle \geq 0$.

There are several alternate definitions of *positive* that are equivalent. Here are two.

Definition 28.5 (equivalent definition of *positive*)**.** A matrix $M \in \mathbb{C}^{d\times d}$ is *positive* if and only if M is normal and all of its eigenvalues are in $\mathbb{R}$ and greater than or equal to 0.

Definition 28.6 (equivalent definition of *positive*)**.** A matrix $M \in \mathbb{C}^{d\times d}$ is *positive* if and only if $M = LL^*$ for some matrix $L \in \mathbb{C}^{d\times d'}$.

The *trace* of a square matrix is simple to define, yet very useful in calculations.

Definition 28.7 (trace)**.** The *trace* of a matrix $M \in \mathbb{C}^{d\times d}$ (denoted as $\mathrm{Tr}(M)$) is defined as the sum of its diagonal entries

$$\mathrm{Tr}(M) = \sum_{k=0}^{d-1} M_{k,k} = \sum_{k=0}^{d-1} \langle k| M |k\rangle . \tag{367}$$

Let's review some properties of the trace. An obvious property is that it is linear. That is, for all $A, B \in \mathbb{C}^{d\times d}$ and all $\alpha, \beta \in \mathbb{C}$,

$$\mathrm{Tr}(\alpha A + \beta B) = \alpha \,\mathrm{Tr}(A) + \beta \,\mathrm{Tr}(B). \tag{368}$$

Also, for all $A \in \mathbb{C}^{c\times d}$ and $B \in \mathbb{C}^{d\times c}$,

$$\mathrm{Tr}(AB) = \mathrm{Tr}(BA). \tag{369}$$

Equation (369) implies that the trace is coordinate system independent, in the sense that, for all $S, A \in \mathbb{C}^{d\times d}$ where S is invertible,

$$\mathrm{Tr}(S^{-1}AS) = \mathrm{Tr}(A). \tag{370}$$

Also, notice that, in Eq. (369), A and B need not be square matrices. For example,

$$\mathrm{Tr}(|\psi\rangle \langle\phi|) = \mathrm{Tr}(\langle\phi| |\psi\rangle) = \langle\phi|\psi\rangle . \tag{371}$$

A word of caution: In general, the trace does *not* have these properties:

- In general, the trace is not multiplicative (in the sense that the determinant is). In general, $\mathrm{Tr}(AB) \neq \mathrm{Tr}(A)\,\mathrm{Tr}(B)$.
- Moreover, Eq. (369) does not mean that you can arbitrarily reorder any product in the argument of the trace. For example, in general, $\mathrm{Tr}(ABC) \neq \mathrm{Tr}(BAC)$. But, for the trace of a product, the product can always be *cyclically* permuted as $\mathrm{Tr}(A_1A_2\ldots A_{m-1}A_m) = \mathrm{Tr}(A_mA_1A_2\ldots A_{m-1})$.

28.5 Characterizing density matrices

Not all matrices arise as the density matrix of some probabilistic mixture of states. The following theorem precisely characterizes which matrices are density matrices.

Theorem 28.3 (characterization of valid density matrix)**.** *A matrix $\rho \in \mathbb{C}^{d\times d}$ is the density matrix of some probabilistic mixture of pure states if and only if ρ is positive and $\operatorname{Tr}(\rho) = 1$.*

Proof. For the "if" part, assume that ρ is positive and $\operatorname{Tr}(\rho) = 1$. By the spectral theorem, $\rho = UDU^*$ for a diagonal D and unitary U. Let $|\phi_0\rangle, |\phi_1\rangle, \ldots, |\phi_{d-1}\rangle$ be the columns of U (which are orthonormal), and $p_0, p_1, \ldots, p_{d-1}$ be the diagonal elements of D. Then we can write

$$\rho = \sum_{j=0}^{d-1} \Big(|\phi_j\rangle\Big)\Big(p_j\Big)\Big(\langle\phi_j|\Big) = \sum_{j=0}^{d-1} p_j \,|\phi_j\rangle\langle\phi_j| \,. \tag{372}$$

Since positivity and unit trace imply that $p_0, p_1, \ldots, p_{d-1}$ is a probability distribution, it follows that this corresponds to a probabilistic mixture.

For the "only if" part, assume that ρ arises from a probabilistic mixture of states

$$\rho = \sum_{k=0}^{r-1} p_k \,|\nu_k\rangle\langle\nu_k| \,, \tag{373}$$

where $|\nu_0\rangle, |\nu_1\rangle, \ldots, |\nu_{r-1}\rangle$ are unit vectors, but they need not be orthogonal. Then, for all $|\psi\rangle$,

$$\begin{aligned}
\langle\psi|\,\rho\,|\psi\rangle &= \langle\psi| \left(\sum_{k=0}^{r-1} p_k \,|\nu_k\rangle\langle\nu_k|\right) |\psi\rangle && (374)\\
&= \sum_{k=0}^{r-1} p_j \,\langle\psi|\nu_k\rangle\langle\nu_k|\psi\rangle && (375)\\
&= \sum_{k=0}^{r-1} p_k |\langle\psi|\nu_k\rangle|^2 && (376)\\
&\geq 0, && (377)
\end{aligned}$$

which implies that ρ is positive.

Also,

$$\mathrm{Tr}(\rho) = \sum_{j=0}^{d-1} \langle j| \left(\sum_{k=0}^{r-1} p_k |\nu_k\rangle \langle \nu_k| \right) |j\rangle \tag{378}$$

$$= \sum_{j=0}^{d-1} \sum_{k=0}^{r-1} p_k \langle j|\nu_k\rangle \langle \nu_k|j\rangle \tag{379}$$

$$= \sum_{k=0}^{r-1} p_k \left(\sum_{j=0}^{d-1} |\langle \nu_k|j\rangle|^2 \right) \tag{380}$$

$$= \sum_{k=0}^{r-1} p_k \tag{381}$$

$$= 1, \tag{382}$$

where Eq. (381) follows from $\sum_{j=0}^{d-1} |\langle \nu_k|j\rangle|^2 = 1$, which is the fact that the sum of the squares of the absolute values of the components of a unit vector is 1. ■

Theorem 28.4 (characterization of pure states)**.** *If $\rho \in \mathbb{C}^{d\times d}$ is a density matrix then ρ is a pure state if and only if* $\mathrm{Tr}(\rho^2) = 1$.

Exercise 28.3. Prove Theorem 28.4.

28.6 Bloch sphere for qubits

The set of density matrices for qubits has a nice representation as points in the *Bloch sphere*. In this section, I explain this correspondence. Consider the Pauli matrices

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \tag{383}$$

(where in this context I is an honorary Pauli matrix). Every 2×2 matrix can be expressed as a linear combination of I, X, Y, Z. Since $\mathrm{Tr}(I) = 2$ and $\mathrm{Tr}(X) = \mathrm{Tr}(Y) = \mathrm{Tr}(Z) = 0$, we can express any density matrix ρ as

$$\rho = \frac{I + c_x X + c_y Y + c_z Z}{2}. \tag{384}$$

Let's develop a geometric picture for the set of all possible triples (c_x, c_y, c_z) that correspond to valid 2×2 density matrices. Let's start with pure states. Any pure state of a qubit can be written as

$$|\psi\rangle = \cos(\tfrac{\theta}{2}) |0\rangle + e^{i\phi} \sin(\tfrac{\theta}{2}) |1\rangle , \tag{385}$$

for some $\theta, \phi \in [0, 2\pi]$.

The density matrix of this state is

$$|\psi\rangle\langle\psi| = \begin{bmatrix} \cos^2(\frac{\theta}{2}) & e^{-i\phi}\cos(\frac{\theta}{2})\sin(\frac{\theta}{2}) \\ e^{i\phi}\cos(\frac{\theta}{2})\sin(\frac{\theta}{2}) & \sin^2(\frac{\theta}{2}) \end{bmatrix} \tag{386}$$

$$= \frac{1}{2}\begin{bmatrix} 1+\cos(\theta) & e^{-i\phi}\sin(\theta) \\ e^{i\phi}\sin(\theta) & 1-\cos(\theta) \end{bmatrix} \tag{387}$$

$$= \frac{1}{2}\begin{bmatrix} 1+\cos(\theta) & \Big(\cos(\phi) - i\sin(\phi)\Big)\sin(\theta) \\ \Big(\cos(\phi) + i\sin(\phi)\Big)\sin(\theta) & 1-\cos(\theta) \end{bmatrix} \tag{388}$$

$$= \frac{I + \cos(\phi)\sin(\theta)X + \sin(\phi)\sin(\theta)Y + \cos(\theta)Z}{2}. \tag{389}$$

Therefore, the coefficients in Eq. (384) are

$$(c_x, c_y, c_z) = (\cos(\phi)\sin(\theta),\ \sin(\phi)\sin(\theta),\ \cos(\theta)). \tag{390}$$

These triples are the *polar coordinates* of points on the surface of a sphere, as shown in figure 202.

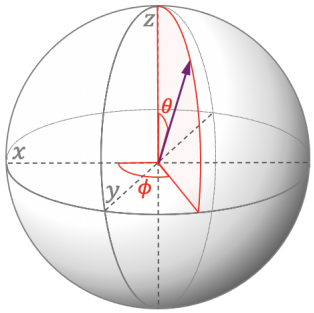

Figure 202: The coordinates (c_x, c_y, c_z) of a pure state are a point on the surface of a sphere.

Think of this sphere as the Earth with the North Pole at the top. Points on the surface can be expressed in terms of their latitude and longitude. The *latitude* θ is the angular distance away from the North Pole. The *longitude* ϕ is an angle representing the East-West distance from some arbitrary[43] starting point. So all the pure states correspond to points[44] on the surface of this sphere, called the *Bloch sphere*.

[43] In geography, the convention is to set 0° at the *Prime Meridian* in Greenwich, UK.

[44] To avoid redundancy, it is natural to restrict the range of θ to $[0, \pi]$.

Where are states $|0\rangle$ and $|1\rangle$ situated on the Bloch sphere? State $|0\rangle$ lies at the North Pole and $|1\rangle$ is at the South Pole, as illustrated in figure 203.

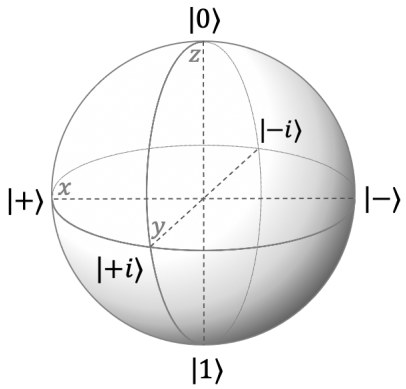

Figure 203: Position of states $|0\rangle$, $|1\rangle$, $|+\rangle$, $|-\rangle$, $|+i\rangle$, and $|-i\rangle$ on the Bloch sphere.

Notice that $|0\rangle$ and $|1\rangle$ are orthogonal as vectors; however, their positions on the Bloch sphere are 180° apart. Any two orthogonal vectors map to antipodal points on the sphere (180° apart).

Where are $|+\rangle$ and $|-\rangle$? They lie on the equator. State $|+\rangle$ has longitude $\phi = 0$ (it lies at the intersection of the equator and the Prime Meridian) and $|-\rangle$ is at the antipodal point.

Notice the angle-doubling again. The angle between $|0\rangle$ and $|+\rangle$ is 45°, but the angle between their points on the Bloch sphere is 90°. In general, for any two state vectors, if we map them to the sphere, the angular distance between them doubles.

There are two other points on the sphere that are in natural positions relative to the points we have considered so far: those that are 90° from $|0\rangle$, $|1\rangle$, $|+\rangle$, and $|-\rangle$. They correspond to the states

$$|+i\rangle = \tfrac{1}{\sqrt{2}}|0\rangle + \tfrac{i}{\sqrt{2}}|1\rangle \tag{391}$$

$$|-i\rangle = \tfrac{1}{\sqrt{2}}|0\rangle - \tfrac{i}{\sqrt{2}}|1\rangle\,. \tag{392}$$

There is some nice symmetry among the six states $|0\rangle$, $|1\rangle$, $|+\rangle$, $|-\rangle$, $|+i\rangle$, and $|-i\rangle$.

The surface of the Bloch sphere consists of all the pure states, and it turns out the *mixed* states are all the points *inside* the sphere. For any probabilistic mixture of pure states, its position in the Bloch sphere is the weighted average of the positions of the pure states. More precisely, suppose that we have an arbitrary 2×2 density matrix ρ and

$$\rho = p_1 |\psi_1\rangle\langle\psi_1| + \cdots + p_m |\psi_m\rangle\langle\psi_m|\,, \tag{393}$$

for pure states $|\psi_1\rangle, \ldots, |\psi_m\rangle$. For each $k \in \{1, \ldots, m\}$, let v_k denote the the point on the surface of the Bloch sphere corresponding to $|\psi_k\rangle\langle\psi_k|$. Then the point in the Bloch sphere corresponding to ρ is

$$p_1 v_1 + \cdots + p_m v_m. \tag{394}$$

For example, an equally weighted mixture of $|0\rangle$ and $|1\rangle$ (the so-called fake-plus state from Eq. (334)) is the point right at the centre of the sphere. And an equally weighted mixture of $|0\rangle$ and $|+\rangle$ is the midpoint of the line connecting their positions on the sphere.

For qubit systems, it's often useful to think of states—and the operations acting on them—on the Bloch sphere.

A word of caution:

- Do not conflate state vectors with points on the Bloch sphere!
- The Bloch sphere is for one-qubit states. For higher dimensional systems, there's an analogous geometric shape—but it's not a hypersphere. It's shape is somewhat complicated and it doesn't satisfy all the properties that one might expect to hold based on the case of qubits. For example, not all points on the surface of this shape are pure states (some are mixed states). For qutrits, the shape is 8-dimensional.

29 Density matrices on multi-register systems

Consider a two-register system, where the first register is c-dimensional and the second register is d-dimensional. The compound register that consists of both systems is cd-dimensional. Mixed states on the compound register can be represented by density matrices of size $cd \times cd$. But there is a rich structure among the subsystems.

When we describe operational procedures involving the two registers, we will often imagine that the first register is *in Alice's lab* and the second register is *in Bob's lab*.

Figure 204: A c-dimensional register in Alice's lab and a d-dimensional register in Bob's lab.

29.1 Product states

A *product state* on a compound register, consisting of a c-dimensional register combined with a d-dimensional register, is is a $cd \times cd$ density matrix of the form

$$\rho = \sigma \otimes \mu, \tag{395}$$

where σ is a $c \times c$ density matrix (representing the state of the first system) and μ is a $d \times d$ density matrix (representing the state of the second system).

A simple example of such a state (where $c = d = 2$) is $|0\rangle\langle 0| \otimes |0\rangle\langle 0|$, which can also be written as $\Big(|0\rangle \otimes |0\rangle\Big)\Big(\langle 0| \otimes \langle 0|\Big) = |00\rangle\langle 00|$. Another example is the state

$$\begin{bmatrix} \frac{3}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} \end{bmatrix} \otimes \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} = \begin{bmatrix} \frac{3}{8} & 0 & \frac{1}{8} & 0 \\ 0 & \frac{3}{8} & 0 & \frac{1}{8} \\ \frac{1}{8} & 0 & \frac{1}{8} & 0 \\ 0 & \frac{1}{8} & 0 & \frac{1}{8} \end{bmatrix}. \tag{396}$$

Clearly a product state among Alice and Bob's labs can be created by having each of them independently create a state within their separate labs. They need no communication whatsoever to do this.

29.2 Separable states

A *separable* state on a compound register, consisting of a c-dimensional register combined with a d-dimensional register, is a $cd \times cd$ density matrix of the form

$$\rho = \sum_{k=1}^{r} p_k \Big(\sigma_k \otimes \mu_k \Big), \tag{397}$$

where $(p_1, \ldots, p_r)$ is a probability vector, $\sigma_1, \ldots, \sigma_r$ are $c \times c$ density matrices, and $\mu_1, \ldots, \mu_r$ are $d \times d$ density matrices.

If the first system is in Alice's lab and the second system is in Bob's lab then the state corresponds to the probabilistic mixture of product states

$$\begin{cases} \sigma_1 \otimes \mu_1 & \text{with prob. } p_1 \\ \sigma_2 \otimes \mu_2 & \text{with prob. } p_2 \\ \quad \vdots & \quad \vdots \qquad \vdots \\ \sigma_r \otimes \mu_r & \text{with prob. } p_r. \end{cases} \tag{398}$$

There are natural ways of preparing this state with only classical communication. One preparation procedure is for Alice to sample a random $k \in \{1, 2, \ldots, r\}$ according to the probability distribution $(p_1, p_2, \ldots, p_r)$ and send k (classical information) to Bob. At this point, Alice can prepare σ_k in her lab and Bob can prepare μ_k in his lab. This shows that a separable state can be prepared *without any quantum communication*. Another procedure is for Bob to sample k and send its value to Alice. Yet another procedure is for a third party to sample k and then send this information to both Alice and Bob.

An example of a separable state that is not a product state is

$$\tfrac{1}{2} |0\rangle\langle 0| \otimes |0\rangle\langle 0| + \tfrac{1}{2} |1\rangle\langle 1| \otimes |1\rangle\langle 1| = \begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}, \tag{399}$$

which corresponds to the probabilistic mixture

$$\begin{cases} |00\rangle & \text{with probability } \frac{1}{2} \\ |11\rangle & \text{with probability } \frac{1}{2}. \end{cases} \tag{400}$$

Exercise 29.1. How do we know that the density matrix in Eq. (399) is not that of a product state? (This is at least worth thinking about intuitively; a formal proof would be to show that it cannot be expressed in the form of Eq. (395).)

29.3 Entangled states

A state is defined to be *entangled* if it is not separable.

The Bell state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$, whose density matrix is

$$\left(\tfrac{1}{\sqrt{2}}|00\rangle + \tfrac{1}{\sqrt{2}}|11\rangle\right)\left(\tfrac{1}{\sqrt{2}}\langle 00| + \tfrac{1}{\sqrt{2}}\langle 11|\right) = \begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{bmatrix}, \tag{401}$$

is an example of a state that is not separable, hence it is entangled (all Bell states are entangled). How do we know it's not separable? We'll address this later on, in section 31.3, where we'll see a nice way of proving that Bell states are not separable.

In general, determining whether a state is separable or not can be difficult. One reason for this is the fact that probabilistic mixtures of states are not unique. For example, consider the 2-qubit state whose density matrix is

$$\rho = \begin{bmatrix} \frac{1}{4} & 0 & 0 & \frac{1}{4} \\ 0 & \frac{1}{4} & \frac{1}{4} & 0 \\ 0 & \frac{1}{4} & \frac{1}{4} & 0 \\ \frac{1}{4} & 0 & 0 & \frac{1}{4} \end{bmatrix}. \tag{402}$$

Is this state separable or entangled? Since ρ is the density matrix of the mixture

$$\begin{cases} \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle & \text{with probability } \frac{1}{2} \\ \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle & \text{with probability } \frac{1}{2}, \end{cases} \tag{403}$$

an operational way of thinking about ρ is that Alice and Bob somehow obtain a random bit and then generate $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ if the bit is 0; and $\frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle$ if the bit is 1. In either case, the state that they generate is entangled (thus not a product state). So does this imply that the state is entangled?

No. In fact, ρ is a separable state. The reason why is that there is *another* probabilistic mixture that also generates the density matrix ρ, namely

$$\begin{cases} |+\rangle|+\rangle & \text{with probability } \frac{1}{2} \\ |-\rangle|-\rangle & \text{with probability } \frac{1}{2}, \end{cases} \tag{404}$$

and the states $|+\rangle|+\rangle$ and $|-\rangle|-\rangle$ (whose density matrices are $|+\rangle\langle+| \otimes |+\rangle\langle+|$ and $|-\rangle\langle-| \otimes |-\rangle\langle-|$, respectively) are product states.

Exercise 29.2 (should be easy)**.** Show that ρ, as defined in Eq. (402), is the density matrix of the probabilistic mixture in Eq. (404).

29.4 The identity $\big(I \otimes |\phi\rangle\big)\,|\psi\rangle = |\psi\rangle \otimes |\phi\rangle$

Section 9 in *Part I: a primer for beginners* began with a simple example of an isometry that can be described operationally as *adding a second qubit in state* $|0\rangle$. This mapping $|\psi\rangle \mapsto |\psi\rangle \otimes |0\rangle$ is represented by a 4×2 matrix. It was not mentioned in section 9 that this matrix can be expressed as the tensor product

$$\begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = I_2 \otimes |0\rangle\,. \tag{405}$$

What if the input state is c-dimensional and the operation is to add some fixed d-dimensional pure state $|\phi\rangle$ after the input state? The $cd \times c$ matrix corresponding to this is $I_c \otimes |\phi\rangle$, where I_c is the $c \times c$ identity matrix. Let's carefully check this.

First of all, $I_c \otimes |\phi\rangle$ is an isometry because

$$\begin{aligned} \big(I_c \otimes |\phi\rangle\big)^* \big(I_c \otimes |\phi\rangle\big) &= \big(I_c \otimes \langle\phi|\big)\big(I_c \otimes |\phi\rangle\big) && (406)\\ &= I_c \otimes I_1 && (407)\\ &= I_c. && (408) \end{aligned}$$

To be clear about the steps, at Eq. (407), we are using the multiplicative property of tensors (Lemma 6.1) and equating the scaler 1 with the 1×1 identity matrix I_1. Also, at Eq. (408), we are using the trivial fact that $M \otimes I_1 = M$, for any matrix M.

The following lemma implies that the matrix $I_c \otimes |\phi\rangle$ corresponds to the mapping $|\psi\rangle \mapsto |\psi\rangle \otimes |\phi\rangle$. The lemma is of independent interest because Eq. (409) arises in calculations; it is worth remembering.

Lemma 29.1. *For all c-dimensional states $|\psi\rangle$ and d-dimensional states $|\phi\rangle$,*

$$\big(I_c \otimes |\phi\rangle\big)\,|\psi\rangle = |\psi\rangle \otimes |\phi\rangle\,. \tag{409}$$

Proof.

$$\begin{aligned} \big(I_c \otimes |\phi\rangle\big)\,|\psi\rangle &= \big(I_c \otimes |\phi\rangle\big)\big(|\psi\rangle \otimes I_1\big) && (410)\\ &= \big(I_c\,|\psi\rangle\big) \otimes \big(|\phi\rangle\, I_1\big) && (411)\\ &= |\psi\rangle \otimes |\phi\rangle\,, && (412) \end{aligned}$$

where Eq. (411) uses the multiplicative property of tensors (Lemma 6.1). ■

29.5 Quantum states of subsystems and the partial trace

Suppose that ρ is a $cd \times cd$ density matrix that represents the joint state of a c-dimensional system in Alice's lab and a d-dimensional system in Bob's lab. Is there a $c \times c$ density matrix σ, which represents the state of Alice's c-dimensional system alone?

Figure 205: Alice's c-dimensional state (ignoring Bob's part of the global state).

In the special case where ρ is a product state of the form $\rho = \sigma \otimes \mu$ it is natural to consider the state of the first system to be σ. Similarly, for the special case of separable states, of the form of Eq. (397), it is natural to think of the state of the first system as

$$\rho = \sum_{k=1}^{r} p_k \sigma_k, \tag{413}$$

since the procedure that Alice performs to construct her state is to sample $\sigma_1, \ldots, \sigma_r$ according to the distribution $(p_1, \ldots, p_r)$.

But what about states that are not separable? Does there *always* exist a density matrix that captures the state of the first system? In the context of *pure states*, this question was discussed back in section 6.3 of *Part I: a primer for beginners* and it turned out that, for entangled states such as the Bell state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$, the subsystems do *not* have pure states of their own (stated in exercise 6.2).

We shall soon see that, in the context of the increased expressive power of *mixed states*, subsystems *always* have states of their own. But, before doing that, let's digress to consider the analogous situation in the context of classical probabilistic states.

29.5.1 Aside: marginal distributions

Suppose that Alice is in possession of $a \in \{1, \ldots, c\}$ and Bob is in possession of $b \in \{1, \ldots, d\}$ and the joint distribution of their states is given by

$$\Pr[\text{ Alice and Bob's joint state is } (a, b)\,] = p_{ab}, \tag{414}$$

where the bivariate probability distribution is often written as a $c \times d$ matrix

$$\begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1d} \\ p_{21} & p_{22} & \cdots & p_{1d} \\ \vdots & \vdots & \ddots & \vdots \\ p_{c1} & p_{c2} & \cdots & p_{cd} \end{bmatrix} \tag{415}$$

and need not be a product distribution. If we want the probability distribution of Alice's state alone, we can simply compute the marginal distribution, consisting of the sums the probabilities in each row of Eq. (415). Namely,

$$\Pr[\,\text{Alice's state is } a\,] = \sum_{b=1}^{d} \Pr[\,\text{Alice and Bob's joint state is } (a,b)\,] \tag{416}$$

$$= \sum_{b=1}^{d} p_{ab}. \tag{417}$$

29.5.2 Quantum analogue of marginal distributions

Returning to our context, Alice and Bob's joint state is ρ, a quantum analogue of a bivariate distribution, and we are seeking a quantum analogue of the marginal distribution corresponding to the state of Alice's system.

One way of deriving the state of Alice's system is to use the fact that what happens to Bob's system is inconsequential, provided that there is no longer any communication whatsoever from Bob to Alice—that Bob essentially *ghosts* Alice. This is alluded to in section 8.3 of *Part I*. Suppose that Alice's system is measured with respect to some orthonormal basis $|\phi_0\rangle, |\phi_1\rangle, \ldots, |\phi_{c-1}\rangle$. To determine the outcome probabilities, make the hypothetical assumptions that:

- Bob also measures his system with respect to the basis, $|0\rangle, |1\rangle, \ldots, |d-1\rangle$.
- There is no communication whatsoever from Bob to Alice.

Under these assumptions, Bob's measurement has no effect on Alice's outcome probabilities, but it becomes easier to calculate them. We can first consider the outcome probabilities for the joint system of Alice and Bob where the joint measurement is with respect to the orthonormal basis $\{|\phi_k\rangle \otimes |\ell\rangle : k \in \{0,1,\ldots,c-1\}, \ell \in \{0,1,\ldots,d-1\}\}$, and the outcome probabilities are

$$\Pr[\,\text{Alice and Bob's joint outcome is } (k,\ell)\,] = \Big(\langle\phi_k| \otimes \langle\ell|\Big)\rho\Big(|\phi_k\rangle \otimes |\ell\rangle\Big). \tag{418}$$

If we want the distribution of Alice's part of the measurement outcome only, we can take the marginal distribution of the joint outcome as

$$\Pr[\text{ Alice's outcome is } k\,] = \sum_{\ell=0}^{d-1} \Pr[\text{ Alice and Bob's joint outcome is } (k,\ell)\,] \tag{419}$$

$$= \sum_{\ell=0}^{d-1} \Big(\langle\phi_k| \otimes \langle\ell|\Big)\rho\Big(|\phi_k\rangle \otimes |\ell\rangle\Big) \tag{420}$$

$$= \langle\phi_k| \left(\sum_{\ell=0}^{d-1} \Big(I_c \otimes \langle\ell|\Big)\rho\Big(I_c \otimes |\ell\rangle\Big)\right) |\phi_k\rangle\,, \tag{421}$$

where at Eq. (421), we are using the fact that $\Big(I_c \otimes |\ell\rangle\Big)\,|\phi_k\rangle = |\phi_k\rangle \otimes |\ell\rangle$, which is proven in Lemma 29.1.

Now, consider the expression in the large parenthesis in Eq. (421), which we refer to as

$$\sigma = \sum_{\ell=0}^{d-1} \Big(I_c \otimes \langle\ell|\Big)\rho\Big(I_c \otimes |\ell\rangle\Big). \tag{422}$$

This is a good candidate for the state of Alice's system because, from Eq. (421), the measurement outcome probabilities for Alice's system are $\langle\phi_k|\,\sigma\,|\phi_k\rangle$.

It is straightforward to check that σ is a valid density matrix. Moreover, σ is the only $c \times c$ density matrix whose outcome probabilities arising from measuring Alice's system with respect to any orthonormal basis $|\phi_0\rangle, |\phi_1\rangle, \ldots, |\phi_{c-1}\rangle$ are $\langle\phi_0|\,\sigma\,|\phi_0\rangle, \langle\phi_1|\,\sigma\,|\phi_1\rangle, \ldots, \langle\phi_{c-1}|\,\sigma\,|\phi_{c-1}\rangle$.

This is our justification for defining the state of the first system of the bipartite state $\rho \in \mathbb{C}^{c\times c} \otimes \mathbb{C}^{d\times d}$ as the expression in Eq. (422). The operation that maps ρ to σ, via Eq. (422), is called the *partial trace*.

29.5.3 Definition of the partial trace

The following definition is in the context of a two-register system, where the first register is c-dimensional and the second register is d-dimensional.

Definition 29.1 (partial trace)**.** Define the operation of *tracing out the second system* as $\mathrm{Tr}_2 : \mathbb{C}^{c\times c} \otimes \mathbb{C}^{d\times d} \to \mathbb{C}^{c\times c}$ where

$$\mathrm{Tr}_2(\rho) = \sum_{\ell=0}^{d-1} \Big(I_c \otimes \langle\ell|\Big)\rho\Big(I_c \otimes |\ell\rangle\Big). \tag{423}$$

The subscript of Tr_2 indicates which register is being traced out. Along similar lines, define the operation of *tracing out the first system* as $\mathrm{Tr}_1 : \mathbb{C}^{c\times c} \otimes \mathbb{C}^{d\times d} \to \mathbb{C}^{d\times d}$ where

$$\mathrm{Tr}_1(\rho) = \sum_{\ell=0}^{c-1} \Big(\langle \ell | \otimes I_d \Big) \rho \Big(|\ell\rangle \otimes I_d \Big). \tag{424}$$

Recall that the *trace* of a square matrix is the sum of its diagonal entries. We can also call this the *full trace* and its definition can be written as $\mathrm{Tr}(\rho) = \sum_{k=0}^{d-1} \langle k | \rho | k \rangle$. The entries of the partial trace of ρ are also sums of the matrix entries of ρ. For example, for the case of two 1-qubit registers,

$$\mathrm{Tr}_1 \begin{bmatrix} \rho_{00,00} & \rho_{00,01} & \rho_{00,10} & \rho_{00,11} \\ \rho_{01,00} & \rho_{01,01} & \rho_{01,10} & \rho_{01,11} \\ \rho_{10,00} & \rho_{10,01} & \rho_{10,10} & \rho_{10,11} \\ \rho_{11,00} & \rho_{11,01} & \rho_{11,10} & \rho_{11,11} \end{bmatrix} = \begin{bmatrix} \rho_{00,00} + \rho_{10,10} & \rho_{00,01} + \rho_{10,11} \\ \rho_{01,00} + \rho_{11,10} & \rho_{01,01} + \rho_{11,11} \end{bmatrix} \tag{425}$$

$$\mathrm{Tr}_2 \begin{bmatrix} \rho_{00,00} & \rho_{00,01} & \rho_{00,10} & \rho_{00,11} \\ \rho_{01,00} & \rho_{01,01} & \rho_{01,10} & \rho_{01,11} \\ \rho_{10,00} & \rho_{10,01} & \rho_{10,10} & \rho_{10,11} \\ \rho_{11,00} & \rho_{11,01} & \rho_{11,10} & \rho_{11,11} \end{bmatrix} = \begin{bmatrix} \rho_{00,00} + \rho_{01,01} & \rho_{00,10} + \rho_{01,11} \\ \rho_{10,00} + \rho_{11,01} & \rho_{10,10} + \rho_{11,11} \end{bmatrix}. \tag{426}$$

It should be noted that, although a measurement was introduced to derive the formulas for the partial trace in section 29.5.2, the measurement does not have to occur. If one register is discarded then the state of the other register is given by the formula for the partial trace whether or not the discarded register is measured.

An alternative way of deriving the formula for $\mathrm{Tr}_2 : \mathbb{C}^{c\times c} \otimes \mathbb{C}^{d\times d} \to \mathbb{C}^{c\times c}$ is to define Tr_2 as the unique linear operator with the property that, for all $A \in \mathbb{C}^{c\times c}$ and $B \in \mathbb{C}^{d\times d}$,

$$\mathrm{Tr}_2(A \otimes B) = A \,\mathrm{Tr}(B). \tag{427}$$

In particular, for density matrices $\sigma \in \mathbb{C}^{c\times c}$ and $\mu \in \mathbb{C}^{d\times d}$, $\mathrm{Tr}_2(\sigma \otimes \mu) = \sigma$.

Now, let's apply the partial trace to calculate the state of the second qubit of the Bell state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$. Applying the formula in Eq. (425) to the density matrix

of the state, we obtain

$$\mathrm{Tr}_1\Big(\big(\tfrac{1}{\sqrt{2}}\left|00\right\rangle + \tfrac{1}{\sqrt{2}}\left|11\right\rangle\big)\big(\tfrac{1}{\sqrt{2}}\left\langle 00\right| + \tfrac{1}{\sqrt{2}}\left\langle 11\right|\big)\Big) = \mathrm{Tr}_1\begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{bmatrix} \tag{428}$$

$$= \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}. \tag{429}$$

There's something remarkable about this. Until now, all our mixed states have been assumed be the result of some probabilistic mixture of pure states. However, a mixed state can arise from a process without any explicit occurrence of randomness or measurement. For the pure state $\frac{1}{\sqrt{2}}\left|00\right\rangle + \frac{1}{\sqrt{2}}\left|11\right\rangle$, the state of each of its individual qubits is $\begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$.

29.6 Purifications

Next, I'd like to show you what a *purification* of a quantum state is. Any mixed state can be viewed as a pure state on some larger system with part of that larger system traced out.

Let ρ be any density matrix of a d-dimensional system. Suppose ρ can be written as a probabilistic mixture of r pure states, as

$$\rho = \sum_{k=0}^{r-1} p_k \left|\nu_k\right\rangle\left\langle\nu_k\right|. \tag{430}$$

Now, consider the pure state on a d-dimensional register and an r-dimensional register

$$\left|\phi\right\rangle = \sum_{k=0}^{r-1} \sqrt{p_k}\left|\nu_k\right\rangle \otimes \left|k\right\rangle. \tag{431}$$

It's easy to see that the state of the first register (namely, with the second system traced out) is $\mathrm{Tr}_2 \left|\phi\right\rangle\left\langle\phi\right| = \rho$. (Remember that, for the purposes of calculation, one can assume that the traced out system is measured.)

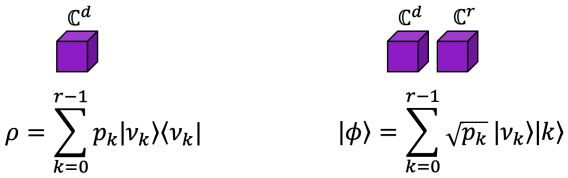

Figure 206: $\left|\phi\right\rangle$ is a purification of the state ρ.

The pure state $|\phi\rangle$ is called a *purification* of ρ. The above shows one way to purify ρ, but the purification of a state is not unique.

In Eq. (430), it is possible that r is much larger than d, in which case the dimension of the second register in the pure state of Eq. (431) is much larger than the first register. However, it is always possible to find a purification of a d-dimensional state where the additional register is also d-dimensional.

Exercise 29.3 (straightforward)**.** Show that, for any $\rho \in \mathbb{C}^{d\times d}$, there exists a purification for which the dimension of the ancillary state is d. In other words, that there exists a purification of the form $|\phi\rangle \in \mathbb{C}^d \otimes \mathbb{C}^d$.

30 State transitions in the Kraus form

So far, we have seen two types of operations that can be performed on quantum systems: isometries and projective measurements. The Kraus form is a unified framework for describing all of these, as well as some other kinds of quantum operations.

Definition 30.1 (Kraus operators)**.** A sequence of $c \times d$ matrices $A_0, A_2, \ldots, A_{m-1}$ are *Kraus operators* if

$$\sum_{k=0}^{m-1} A_k^* A_k = I_d. \tag{432}$$

Note that the matrices need not be square. If A_k is $c \times d$ then $A_k^* A_k$ is $d \times d$.

At first glance, Definition 30.1 may look mysterious. In this section, we'll see that several natural transformations are expressible in terms of Kraus operators.

30.1 Measurements via Kraus operators

For a sequence of Kraus operators $A_0, A_2, \ldots, A_{m-1} \in \mathbb{C}^{c \times d}$, define the following measurement operation, whose classical output is $k \in \{0, 1, \ldots, m-1\}$.

Input to the measurement: a d-dimensional quantum system in state $\rho \in \mathbb{C}^{d \times d}$.

Output of the measurement: the probabilistic mixture

$$\begin{cases} \left(\quad 0\,, \quad \dfrac{A_0 \rho A_0^*}{\mathrm{Tr}(A_0 \rho A_0^*)} \quad \right) & \text{with prob. } \mathrm{Tr}(A_0 \rho A_0^*) \\ \left(\quad 1\,, \quad \dfrac{A_1 \rho A_1^*}{\mathrm{Tr}(A_1 \rho A_1^*)} \quad \right) & \text{with prob. } \mathrm{Tr}(A_1 \rho A_1^*) \\ \quad \vdots \qquad\qquad \vdots & \qquad\quad \vdots \qquad\quad \vdots \\ \left(m-1\,, \quad \dfrac{A_{m-1} \rho A_{m-1}^*}{\mathrm{Tr}(A_{m-1} \rho A_{m-1}^*)} \right) & \text{with prob. } \mathrm{Tr}(A_{m-1} \rho A_{m-1}^*). \end{cases} \tag{433}$$

In other words, for input state $\rho \in \mathbb{C}^{d \times d}$, the classical part of the measurement outcome is $k \in \{0, 1, \ldots, m-1\}$, where

$$\Pr[\text{classical outcome is } k] = \mathrm{Tr}(A_k \rho A_k^*), \tag{434}$$

and the corresponding residual state is

$$\frac{A_k \rho A_k^*}{\mathrm{Tr}(A_k \rho A_k^*)}. \tag{435}$$

Let's first check that the above makes sense. Are the probabilities non-negative real numbers that sum to 1? Are the residual states valid density matrices? To get an idea of why this measurement makes sense, let us first consider the case of pure states. If $\rho = |\psi\rangle\langle\psi|$ then, for all k,

$$\mathrm{Tr}(A_k \rho A_k^*) = \mathrm{Tr}(A_k |\psi\rangle\langle\psi| A_k^*) = \mathrm{Tr}(\langle\psi| A_k^* A_k |\psi\rangle) = \langle\psi| A_k^* A_k |\psi\rangle . \tag{436}$$

Clearly, $\langle\psi| A_k^* A_k |\psi\rangle \geq 0$, since this is the inner product of $A_k |\psi\rangle$ with itself. Also, the sum of the outcome probabilities is

$$\sum_{k=0}^{m-1} \mathrm{Tr}(A_k \rho A_k^*) = \sum_{k=0}^{m-1} \langle\psi| A_k^* A_k |\psi\rangle = \langle\psi| \left(\sum_{k=0}^{m-1} A_k^* A_k \right) |\psi\rangle = \langle\psi|\psi\rangle = 1. \tag{437}$$

And the residual state corresponding to classical outcome k is

$$\frac{A_k \rho A_k^*}{\mathrm{Tr}(A_k \rho A_k^*)} = \frac{A_k |\psi\rangle\langle\psi| A_k^*}{\langle\psi| A_k^* A_k |\psi\rangle} \tag{438}$$

$$= \frac{A_k |\psi\rangle\langle\psi| A_k^*}{\| A_k |\psi\rangle \|^2} \tag{439}$$

$$= \left(\frac{A_k |\psi\rangle}{\| A_k |\psi\rangle \|} \right) \left(\frac{\langle\psi| A_k^*}{\| A_k |\psi\rangle \|} \right), \tag{440}$$

which is the density matrix of the (normalized) pure state $A_k |\psi\rangle / \| A_k |\psi\rangle \|$. What if $\| A_k |\psi\rangle \| = 0$? In that case, $\langle\psi| A_k^* A_k |\psi\rangle = 0$, so those outcomes never occur.

To understand the measurement in the case of arbitrary mixed states, we can use the fact that that every d-dimensional state can be expressed as

$$\rho = \sum_{j=0}^{d-1} p_j |\psi_j\rangle\langle\psi_k| \tag{441}$$

and interpret this as a probabilistic mixture of pure states (whether or not the state is actually generated by such a probabilistic process). Then the probability of classical outcome k is the weighted average of the probabilities for the pure states in the mixture

$$\sum_{j=0}^{d-1} p_j \langle\psi_j| A_k^* A_k |\psi_j\rangle , \tag{442}$$

which is clearly non-negative, and these probabilities sum to

$$\sum_{k=0}^{m-1}\sum_{j=0}^{d-1} p_j \langle\psi_j| A_k^* A_k |\psi_j\rangle = \sum_{j=0}^{d-1} p_j \Big(\sum_{k=0}^{m-1} \langle\psi_j| A_k^* A_k |\psi_j\rangle \Big) \tag{443}$$

$$= \sum_{j=0}^{d-1} p_j \tag{444}$$

$$= 1. \tag{445}$$

Also, the residual state corresponding to outcome k is the mixed state expressible as

$$\begin{cases} \dfrac{A_k |\psi_0\rangle}{\| A_k |\psi_0\rangle \|} & \text{with prob. } p_0 \\ \dfrac{A_k |\psi_1\rangle}{\| A_k |\psi_1\rangle \|} & \text{with prob. } p_1 \\ \vdots \qquad\qquad \vdots & \qquad\qquad \vdots \\ \dfrac{A_k |\psi_{d-1}\rangle}{\| A_k |\psi_{d-1}\rangle \|} & \text{with prob. } p_{d-1}. \end{cases} \tag{446}$$

Next, we'll see some measurements expressed in terms of Kraus operators.

30.1.1 Projective measurements

Projective measurements, which are defined in definition 8.3 of section 8, are a special case of Kraus measurements. Let $\Pi_0, \Pi_1, \ldots, \Pi_{m-1} \in \mathbb{C}^{d\times d}$ be a sequence of pairwise-orthogonal and complete projectors. These are valid Kraus operators since

$$\sum_{k=0}^{m-1} \Pi_k^* \Pi_k = \sum_{k=0}^{m-1} (\Pi_k)^2 = \sum_{k=0}^{m-1} \Pi_k = I. \tag{447}$$

Recall these special cases of projective measurements:

- A measurement with respect to an orthonormal basis $|\phi_0\rangle, |\phi_1\rangle, \ldots, |\phi_{d-1}\rangle$ can be expressed as the projective measurement with projectors

$$|\phi_0\rangle\langle\phi_0|, \quad |\phi_1\rangle\langle\phi_1|, \quad \ldots, \quad |\phi_{d-1}\rangle\langle\phi_{d-1}|. \tag{448}$$

- If there are two registers, with dimensions c and d, then a measurement of the first register with respect to the orthonormal basis $|\phi_0\rangle, |\phi_1\rangle, \ldots, |\phi_{c-1}\rangle$ can be expressed as the projective measurement with projectors

$$|\phi_0\rangle\langle\phi_0| \otimes I_d, \quad |\phi_1\rangle\langle\phi_1| \otimes I_d, \quad \ldots, \quad |\phi_{c-1}\rangle\langle\phi_{c-1}| \otimes I_d. \tag{449}$$

But Kraus measurements are much more general than projective measurements.

30.1.2 A Kraus measurement for the trine states

The trine states (which arose as a running example in *Part I*, sections 5.1.2 and 9.2.1) are three vectors in $\mathbb{C}^2$, with angle 120° between each pair

$$|\phi_0\rangle = |0\rangle \tag{450}$$

$$|\phi_1\rangle = -\tfrac{1}{2}|0\rangle + \tfrac{\sqrt{3}}{2}|1\rangle \tag{451}$$

$$|\phi_2\rangle = -\tfrac{1}{2}|0\rangle - \tfrac{\sqrt{3}}{2}|1\rangle\,. \tag{452}$$

Define these three Kraus operators

$$A_0 = \sqrt{\tfrac{2}{3}}\,|\phi_0\rangle\langle\phi_0| \;=\; \begin{bmatrix} \sqrt{2/3} & 0 \\ 0 & 0 \end{bmatrix} \tag{453}$$

$$A_1 = \sqrt{\tfrac{2}{3}}\,|\phi_1\rangle\langle\phi_1| \;=\; \frac{1}{4}\begin{bmatrix} \sqrt{2/3} & -\sqrt{2} \\ -\sqrt{2} & \sqrt{6} \end{bmatrix} \tag{454}$$

$$A_2 = \sqrt{\tfrac{2}{3}}\,|\phi_2\rangle\langle\phi_2| \;=\; \frac{1}{4}\begin{bmatrix} \sqrt{2/3} & \sqrt{2} \\ \sqrt{2} & \sqrt{6} \end{bmatrix}. \tag{455}$$

Notice that these are *not* projectors (because of the factor $\sqrt{2/3}$) and they are not orthogonal. Nevertheless, since $A_0^*A_0 + A_1^*A_1 + A_2^*A_2 = I$, these are valid Kraus operators. It is straightforward to calculate that using this measurement for the trine state distinguishing problem results in success probability $\frac{2}{3}$.

This is an alternative to the "exotic" measurement procedure explained in section 9.2.1, where the 2-dimensional system system is isometrically embedded into a 3-dimensional space and then a projective measurement is applied. With Kraus measurements, there is no need to embed into a larger space.

30.1.3 POVM measurements

Kraus measurements are frequently expressed in a simplified form, called *POVM measurements* (where POVM stands for *positive operator valued measure*[45]). This simplified form makes the most sense when we *only* care about the classical outcome (and not the residual quantum state).

[45] Regarding terminology, a *positive operator valued measure* is a generalization of the notion of a *measure*, that you may have seen in probability theory or functional analysis. The meaning of "measure" is completely distinct from "measurement". So "POVM measurement" is not redundant.

Recall that, for Kraus operators $A_0, A_1, \ldots, A_{m-1}$, the associated measurement of a state ρ produces outcome $k \in \{0, 1, \ldots, m-1\}$ with probability

$$\mathrm{Tr}(A_k \rho A_k^*) = \mathrm{Tr}(\rho\, A_k^* A_k). \tag{456}$$

For each Kraus operator A_k, define $E_k = A_k^* A_k$. All we need to know is the sequence $E_0, E_1, \ldots, E_{m-1}$ to define the classical part of the measurement outcome. And we can characterize such sequences $E_0, E_1, \ldots, E_{m-1}$ in a simple way.

Definition 30.2 (POVM elements)**.** A sequence $E_0, E_1, \ldots, E_{m-1}$ is a *sequence of POVM elements* if, for all k, it holds that E_k is positive,[46] and

$$E_0 + E_1 + \cdots + E_{m-1} = I. \tag{457}$$

For a sequence of POVM elements $E_0, E_1, \ldots, E_{m-1} \in \mathbb{C}^{d \times d}$ and a d-dimensional quantum state ρ, applying the associated *POVM measurement* produces outcome $k \in \{0, 1, \ldots, m-1\}$ with probability $\mathrm{Tr}(\rho E_k)$.

A word of caution: if we care about the residual quantum state after the measurement then POVM measurements are less expressive than Kraus measurements. We can find an A_k such that $E_k = A_k^* A_k$, but this A_k is not unique, and for different choices of A_k the residual state is different.

The POVM measurement elements corresponding to the trine state measurement in section 30.1.2 are

$$E_0 = \tfrac{2}{3} |\phi_0\rangle \langle\phi_0|, \quad E_1 = \tfrac{2}{3} |\phi_1\rangle \langle\phi_1|, \quad E_2 = \tfrac{2}{3} |\phi_2\rangle \langle\phi_2|. \tag{458}$$

30.2 Quantum channels via Kraus operators

For a sequence of Kraus operators, $A_0, A_2, \ldots, A_{m-1} \in \mathbb{C}^{c \times d}$, define the following state transformation, called a *quantum channel*, which maps quantum states to quantum states with no classsical side information.

Input to the channel: is a d-dimensional quantum system, whose state can be described by a $d \times d$ density matrix ρ.

Output of the channel: is a c-dimensional quantum system, whose state is

$$A_0 \rho A_0^* + A_1 \rho A_1^* + \cdots + A_{m-1} \rho A_{m-1}^*. \tag{459}$$

[46] Meaning that E_k is normal and all its eigenvalues are nonnegative reals.

One interpretation of the quantum channel with Kraus operators $A_0, A_2, \ldots, A_{m-1}$ is as a process where the Kraus measurement (as defined in section 30.1) occurs, but the classical part of the outcome is withheld. We can imagine that Bob sends the state to Alice, who performs the measurement (and sees the classical outcome) and then Alice sends the residual quantum state back to Bob, but she does not send him the classical output of the measurement.

Therefore, from Bob's perspective, the quantum state is the probabilistic mixture of states in Eq. (433), but with the classical part of the outcome omitted. This state has density matrix

$$\sum_{k=0}^{m-1} \frac{A_k \rho A_k^*}{\mathrm{Tr}(A_k \rho A_k^*)} \mathrm{Tr}(A_k \rho A_k^*) = \sum_{k=0}^{m-1} A_k \rho A_k^* \tag{460}$$

(any terms where $\mathrm{Tr}(A_k \rho A_k^*) = 0$ correspond to outcomes that never occur and where $A_k \rho A_k^* = 0$). This interpretation is sometimes useful—though there are other fruitful ways of thinking about channels. The subject of quantum channels is quite large, and has an extensive underlying theory.

What follows are a few simple examples of quantum channels.

30.2.1 Unitary operations and isometries

Any $d \times d$ unitary operation U corresponds to a quantum channel with one single Kraus operator U. The channel maps each $d \times d$ density matrix ρ maps to $U \rho U^*$. Similarly, for any $c \times d$ isometry V. So quantum channels generalize isometries.

The examples that follow are channels that have more than one Kraus operator.

30.2.2 Decoherence of a qubit

I will first explain what this channel does, and then show you two different ways of "implementing" the channel in terms of Kraus operators. The decoherence channel changes the state of its input qubit from

$$\rho = \begin{bmatrix} \rho_{00} & \rho_{01} \\ \rho_{10} & \rho_{11} \end{bmatrix} \qquad \text{to} \qquad \begin{bmatrix} \rho_{00} & 0 \\ 0 & \rho_{11} \end{bmatrix}. \tag{461}$$

The diagonal density matrix can be viewed as a probabilistic mixture of $|0\rangle\langle 0|$ and $|1\rangle\langle 1|$. On the Bloch sphere, the diagonal density matrices are on the axis connecting

$|0\rangle\langle 0|$ and $|1\rangle\langle 1|$. The effect of the channel is to move the state "horizontally" (i.e., parallel to the equatorial plane) to the vertical axis connecting $|0\rangle\langle 0|$ and $|1\rangle\langle 1|$.

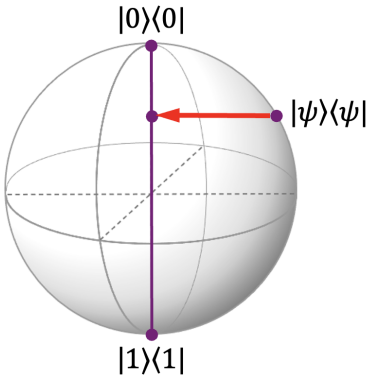

Figure 207: Effect of the decoherence channel on pure state $|\psi\rangle\langle\psi|$ on the Bloch sphere.

Below are two operationally different ways of implementing this channel.

Decoherence by measuring without looking at the outcome

Our first way of implementing the decoherence channel is intuitively based on the operation of measuring the qubit in the computational basis, but without seeing the classical part of the outcome. Imagine Bob sends his qubit to Alice, who measures in the computational basis (and sees the outcome), and then Alice sends the qubit back to Bob, but does not send the classical output of the measurement.

The quantum part of the outcome of Alice's measurement is

$$\begin{cases} |0\rangle & \text{with prob. } \langle 0|\,\rho\,|0\rangle \\ |1\rangle & \text{with prob. } \langle 1|\,\rho\,|1\rangle. \end{cases} \tag{462}$$

Since Alice obtains the classical outcome, from *her* perspective, the quantum outcome is always either $|0\rangle\langle 0|$ or $|1\rangle\langle 1|$. But Bob does not receive the classical outcome so, from *his* perspective, the quantum outcome is the density matrix

$$\langle 0|\rho|0\rangle\,|0\rangle\langle 0| + \langle 1|\rho|1\rangle\,|1\rangle\langle 1|\,. \tag{463}$$

This can be expressed as a channel by setting the Kraus operators to $A_0 = |0\rangle\langle 0|$ and $A_1 = |1\rangle\langle 1|$. Then a density matrix $\rho = \left[\begin{smallmatrix}\rho_{00} & \rho_{01}\\ \rho_{10} & \rho_{11}\end{smallmatrix}\right]$ maps to

$$\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix}\begin{bmatrix}\rho_{00} & \rho_{01}\\ \rho_{10} & \rho_{11}\end{bmatrix}\begin{bmatrix}1 & 0\\ 0 & 0\end{bmatrix} + \begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\rho_{00} & \rho_{01}\\ \rho_{10} & \rho_{11}\end{bmatrix}\begin{bmatrix}0 & 0\\ 0 & 1\end{bmatrix} = \begin{bmatrix}\rho_{00} & 0\\ 0 & \rho_{11}\end{bmatrix}. \tag{464}$$

Decoherence by applying a probabilistic mixture of I and Z

Another way of implementing the decoherence channel is intuitively based on applying a randomly selected unitary to the state. Bob sends the qubit to Alice, who does the following. She flips a fair coin, and then either applies I or Z to the qubit, depending on the outcome of the coin flip. Then she sends the qubit back to Bob, but she does not reveal the coin flip.

Since Alice knows outcome of the coin flip, from *her* perspective, the state is either ρ or $Z\rho Z$. But Bob does not know the coin flip so, from *his* perspective, the state is

$$\begin{cases} \rho & \text{with prob. } \frac{1}{2} \\ Z\rho Z & \text{with prob. } \frac{1}{2}. \end{cases} \tag{465}$$

and the density matrix of this mixture is

$$\tfrac{1}{2}\rho + \tfrac{1}{2}Z\rho Z. \tag{466}$$

This can be expressed in the Kraus form by setting the Kraus operators of a quantum channel to $A_0 = \frac{1}{\sqrt{2}}I$ and $A_1 = \frac{1}{\sqrt{2}}Z$. Then a density matrix $\rho = \left[\begin{smallmatrix} \rho_{00} & \rho_{01} \\ \rho_{10} & \rho_{11} \end{smallmatrix}\right]$ maps to

$$\left(\tfrac{1}{\sqrt{2}}I\right)\rho\left(\tfrac{1}{\sqrt{2}}I\right)^* + \left(\tfrac{1}{\sqrt{2}}Z\right)\rho\left(\tfrac{1}{\sqrt{2}}Z\right)^* = \tfrac{1}{2}\rho + \tfrac{1}{2}Z\rho Z \tag{467}$$

$$= \tfrac{1}{2}\begin{bmatrix} \rho_{00} & \rho_{01} \\ \rho_{10} & \rho_{11} \end{bmatrix} + \tfrac{1}{2}\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\begin{bmatrix} \rho_{00} & \rho_{01} \\ \rho_{10} & \rho_{11} \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{468}$$

$$= \begin{bmatrix} \rho_{00} & 0 \\ 0 & \rho_{11} \end{bmatrix}. \tag{469}$$

Comparison of the two implementations of the decoherence channel

From Bob's perspective, who doesn't receive any classical measurement outcomes or coin flip outcomes, the two implementations of the decoherence channel are identical. However, they are not the same if we include Alice's perspective. In the first implementation, the state is actually measured and it cannot be recovered at a later time, even with the classical information that Alice has. In the second implementation, Alice performs no measurement. If, at some later time, she reveals the coin flip to Bob then he can recover the initial state (by applying either I or Z to the state).

So there's an advantage to the second implementation. But there is also a disadvantage: if Bob asks Alice later on "what was the measurement outcome?", she

cannot answer that question. There is no classical bit $b \in \{0,1\}$ that Alice can produce and send to Bob such that, if Bob then measures his decohered state in the computational basis, the outcome is guaranteed to be b.

Exercise 30.1 (conceptual)**.** Suppose that Bob believes that he has figured out a new way of measuring a qubit that is reversible. His idea is to first implement the random unitary method to create the decohered state, which can serve as the quantum outcome (remembering what the coin flip is, so that the can undo the unitary later on). Now all that's lacking is that classical outcome. Bob's idea is to measure the decohered qubit in the computational basis to obtain a bit that can serve as the the classical outcome of the measurement. Will doing all this result in a faithful simulation of the measurement operation? And, after all these operations have been performed, is there a way for Bob to recover the original state?

General definition of a mixed-unitary channel

This notion of defining a channel as a probabilistic mixture of unitary operations can be defined in general as follows.

Definition 30.3 (mixed-unitary channel)**.** A channel is *mixed-unitary* if it can be expressed by Kraus operators of the form

$$A_k = \sqrt{p_k}\, U_k, \tag{470}$$

for $k \in \{0, 1, \ldots, m-1\}$, where $(p_0, p_1, \ldots, p_{m-1})$ is a valid probability vector and $U_0, U_1, \ldots, U_{m-1} \in \mathbb{C}^{d\times d}$ are unitary.

Such a channel maps $\rho \in \mathbb{C}^{d\times d}$ to

$$p_0\, U_0\, \rho\, U_0^* \;+\; p_1\, U_1\, \rho\, U_1^* \;+\; \cdots \;+\; p_{m-1}\, U_{m-1}\, \rho\, U_{m-1}^*, \tag{471}$$

which can be interpreted operationally as choosing a random k and then applying U_k.

Note that, since the Kraus representation of a channel is not unique, the mere existence of a Kraus representation that is not of the above mixed-unitary form does not imply that the channel is not mixed-unitary; there may be *another* Kraus representation that *is* of the mixed-unitary form. For example, consider the two different Kraus representations of the decoherence channel in section 30.2.2.

We will eventually see an example of channel which takes qubits to qubits that is not mixed-unitary (that has no representation that is of the mixed-unitary form).

There is an obvious generalization of mixed-unitary to *mixed-isometry*, which I do not elaborate on here.

30.2.3 Phase damping channel

This channel maps qubits to qubits and is parameterized by $p \in [0, \frac{1}{2}]$. It corresponds to the probabilistic mixture of unitaries

$$\begin{cases} I & \text{with prob. } 1-p \\ Z & \text{with prob. } p. \end{cases} \tag{472}$$

Kraus operators for this channel are $A_0 = \sqrt{1-p}\, I$ and $A_1 = \sqrt{p}\, Z$.

The effect of the channnel is the mapping

$$\begin{bmatrix} \rho_{00} & \rho_{01} \\ \rho_{10} & \rho_{11} \end{bmatrix} \mapsto \begin{bmatrix} \rho_{00} & (1-2p)\rho_{01} \\ (1-2p)\rho_{10} & \rho_{11} \end{bmatrix}. \tag{473}$$

For the setting $p = 0$, the channel does not change the state at all, and for the setting $p = \frac{1}{2}$, the channel is the decoherence channel of section 30.2.2. For the intermediate values of p, where $0 < p < \frac{1}{2}$, the channel "partially decoheres" the state. On the Bloch sphere, it moves the state part of the way towards the vertical axis between $|0\rangle\langle 0|$ and $|1\rangle\langle 1|$, as illustrated in figure 208. The channel shrinks the Bloch sphere

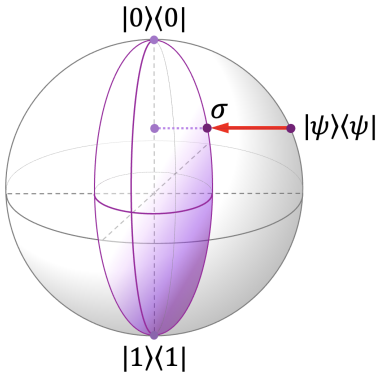

Figure 208: Effect of phase damping channel on pure state $|\psi\rangle\langle\psi|$ on the Bloch sphere.

to an ellipsoid connecting $|0\rangle\langle 0|$ and $|1\rangle\langle 1|$.

30.2.4 Depolarizing channel

This channel is frequently used as a standard qubit noise model (relevant for quantum error-correcting codes, the subject of sections 35 and 36). It is parameterized by

$p \in [0, \frac{3}{4}]$ and corresponds to the probabilistic mixture of unitaries

$$\begin{cases} I & \text{with prob. } 1-p \\ X & \text{with prob. } p/3 \\ Y & \text{with prob. } p/3 \\ Z & \text{with prob. } p/3. \end{cases} \tag{474}$$

Kraus operators for this channel are

$$A_0 = \sqrt{1-p}\, I, \quad A_1 = \sqrt{p/3}\, X, \quad A_2 = \sqrt{p/3}\, Y, \quad A_3 = \sqrt{p/3}\, Z. \tag{475}$$

It can be calculated that the effect of the depolarizing channel is the mapping

$$\rho \;\mapsto\; \frac{4p}{3}\begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} + \Big(1 - \frac{4p}{3}\Big)\rho. \tag{476}$$

When $p > 0$, this corresponds to moving the state towards the state $\frac{1}{2}I$ (at the centre of the Bloch sphere), as illustrated in figure 209. The channel shrinks the Bloch

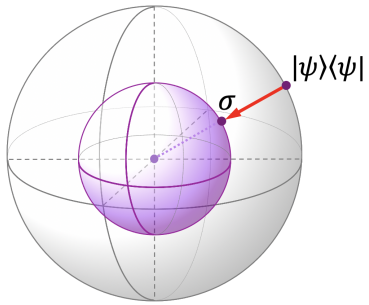

Figure 209: Effect of depolarizing channel on pure state $|\psi\rangle\langle\psi|$ on the Bloch sphere.

sphere symetrically towards the centre.

Exercise 30.2. Prove that the depolarizing channel, as described by Eq. (474), results in the mapping in Eq. (476).

30.2.5 Amplitude damping channel

This channel is not a mixed-unitary channel; it cannot be expressed as a probabilistic mixture of unitaries. It is parameterized by $p \in [0, 1]$, and can be described by Kraus operators

$$A_0 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{bmatrix} \qquad \text{and} \qquad A_1 = \begin{bmatrix} 0 & \sqrt{p} \\ 0 & 0 \end{bmatrix}. \tag{477}$$

It is straightforward to calculate that the effect of the channnel is the mapping

$$\begin{bmatrix} \rho_{00} & \rho_{01} \\ \rho_{10} & \rho_{11} \end{bmatrix} \mapsto \begin{bmatrix} \rho_{00} + p\rho_{11} & \sqrt{1-p}\,\rho_{01} \\ \sqrt{1-p}\,\rho_{10} & (1-p)\rho_{11} \end{bmatrix}. \tag{478}$$

When $p > 0$, this shrinks all entries of the density matrix except ρ_{00}. It moves the state towards $|0\rangle\langle 0|$, as illustrated in figure 210. The channel shrinks the Bloch sphere

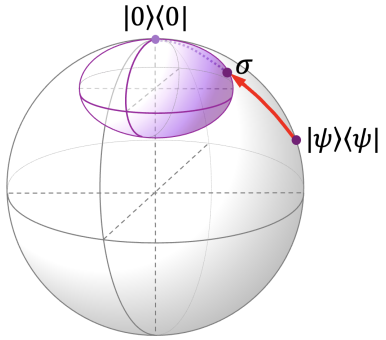

Figure 210: Effect of amplitude damping channel on pure state $|\psi\rangle\langle\psi|$ on the Bloch sphere.

to a vertically flattened ellipsoid, positioned so that $|0\rangle\langle 0|$ remains fixed.

Exercise 30.3. Prove that the amplitude damping channel is not mixed-unitary.

30.2.6 Adding a mixed-state ancilla register

We have seen in section 29.4 that the operation of adding an ancilla register in pure state $|\phi\rangle$ to a c-dimensional state (the mapping $\rho \mapsto \rho \otimes |\phi\rangle\langle\phi|$) corresponds to the isometry $I_c \otimes |\phi\rangle$. Suppose that we want to add an ancilla in some fixed *mixed* state σ? This is the mapping $\rho \mapsto \rho \otimes \sigma$. How do we express this in terms of Kraus operators?

Exercise 30.4. For a fixed d-dimensional mixed state σ, show how to express the mapping $\rho \mapsto \rho \otimes \sigma$ as a quantum channel in the Kraus form.

30.2.7 The partial trace is a channel

We conclude by noting that the partial trace, defined in Definition 29.1, is a channel. If there are two registers, of dimensions c and d, then the operation of *tracing out the second register* (denoted as Tr_2) is expressible by the Kraus operators

$$I_c \otimes \langle 0|\,, \quad I_c \otimes \langle 1|\,, \quad \ldots, \quad I_c \otimes \langle d-1|\,. \tag{479}$$

Operationally, this corresponds to removing the second register, which is kind of complementary to the operation of adding a second register.

31 Channels and linear maps on subsystems

Let A_0, A_1, $\ldots$, A_{m-1} be $c \times d$ Kraus operators, which define a channel χ that maps each $d \times d$ density matrix ρ to the $c \times c$ density matrix

$$\chi(\rho) = \sum_{k=0}^{m-1} A_k \rho A_k^*. \tag{480}$$

Now suppose that there is an additional register in play and the channel doesn't act on that additional register. This scenario is reminiscent to the example in section 6.6 of *Part I*, where we considered a 1-qubit unitary acting on the second qubit of a two-qubit system.

Suppose that our *first* register is r-dimensional (that supports states described as $r \times r$ density matrices) and our *second* register is d-dimensional, and we apply the channel described in Eq. (480) to the second system—and perform no action on the first system. We can define this as a channel acting on the combined system by the $rc \times rd$ Kraus operators

$$I_r \otimes A_0, \;\; I_r \otimes A_1, \;\; \ldots, \;\; I_r \otimes A_{m-1}. \tag{481}$$

The input is an $rd \times rd$ density matrix and the output is an $rc \times rc$ density matrix. Let's denote this channel on the joint system as $I_r \otimes \chi$.

In the special case where input to the channel $I_r \otimes \chi$ is a product state of the form $\rho \otimes \sigma \in \mathbb{C}^{r \times r} \otimes \mathbb{C}^{d \times d}$, the channel acts as

$$(I_r \otimes \chi)(\rho \otimes \sigma) = \rho \otimes \chi(\sigma). \tag{482}$$

Moreover, in the special case where the input to channel $I_r \otimes \chi$ is a separable state (discussed in section 29.2) of the form

$$\sum_{j=1}^{n} p_j \Big(\rho_j \otimes \sigma_j \Big), \tag{483}$$

the channel acts as

$$(I_r \otimes \chi) \left(\sum_{j=1}^{n} p_j \Big(\rho_j \otimes \sigma_j \Big) \right) = \sum_{j=1}^{n} p_j \Big(\rho_j \otimes \chi(\sigma_j) \Big). \tag{484}$$

31.1 Channels vs. linear maps

A map $f : \mathbb{C}^{d\times d} \to \mathbb{C}^{c\times c}$ is *linear* if, for all $\lambda_1, \lambda_2 \in \mathbb{C}$, and for all $A_1, A_2 \in \mathbb{C}^{d\times d}$,

$$f(\lambda A_1 + \lambda_2 A_2) = \lambda_1 f(A_1) + \lambda_2 f(A_2). \tag{485}$$

Note that all channels are linear maps: if $\chi : \mathbb{C}^{d\times d} \to \mathbb{C}^{c\times c}$ is a channel then χ is linear.

Exercise 31.1 (easy)**.** Prove that all channels are linear maps.

However, not all linear maps are channels. There are many trivial counterexamples, which are linear maps that do not map all valid density matrices to valid density matrices, such the zero function, or the function that multiplies all entries of the input density matrix by 2 (so the output matrix would not have trace 1).

There are also more subtle counterexamples, which are mappings that map every valid input density matrix to a valid output density matrix *and nevertheless are not channels*.

Consider the transpose mapping: $\rho \mapsto \rho^T$, where ρ^T is the transpose of matrix ρ. This is linear and has the property that it maps all valid density matrices to valid density matrices (of the same dimension). Is the transpose a valid channel? In fact, it is not, and this will be proved in subsection 31.3.

31.2 Linear maps applied to subsystems

Let $f : \mathbb{C}^{d\times d} \to \mathbb{C}^{c\times c}$ be any linear map (that takes a $d_2 \times d_2$ matrix as input and produces a $d_1 \times d_1$ matrix as output). Suppose that an additional r-dimensional register is introduced and, informally speaking, f "does nothing" to that additional register. Let's not worry about what (if anything) that informal statement means. Instead, let's formally define the linear map $I_r \otimes f$ that maps $rd \times rd$ matrices to $rc \times rc$ matrices as follows. For all $A \in \mathbb{C}^{r\times r}$ and $B \in \mathbb{C}^{d\times d}$,

$$(I_r \otimes f)\Big(A \otimes B\Big) = A \otimes f(B). \tag{486}$$

And this extends by linearity to a mapping from *all* $rd \times rd$ matrices because every $rd \times rd$ matrix M can be expressed as

$$M = \sum_{i=1}^{r}\sum_{j=1}^{r} E_{ij} \otimes A_{ij} = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1c} \\ A_{21} & A_{22} & \cdots & A_{2c} \\ \vdots & \vdots & \ddots & \vdots \\ A_{c1} & A_{c2} & \cdots & A_{cc} \end{bmatrix}, \tag{487}$$

where each A_{ij} is a $d \times d$ matrix (not necessarily a density matrix), and $E_{ij} = |i\rangle\langle j|$, which is the $r \times r$ matrix whose $i'j'$-th entry is

$$\begin{cases} 1 & \text{if } i' = i \text{ and } j' = j \\ 0 & \text{otherwise.} \end{cases} \tag{488}$$

It follows from Eq. (486), the linearity of $I_r \otimes \chi$, and Eq. (487) that

$$(I_r \otimes f)(M) = \sum_{i=1}^{r} \sum_{j=1}^{r} E_{ij} \otimes f(A_{ij}) = \begin{bmatrix} f(A_{11}) & f(A_{12}) & \cdots & f(A_{1r}) \\ f(A_{21}) & f(A_{22}) & \cdots & f(A_{2r}) \\ \vdots & \vdots & \ddots & \vdots \\ f(A_{r1}) & f(A_{r2}) & \cdots & f(A_{rr}) \end{bmatrix}. \tag{489}$$

In particular, in the special case where the linear mapping is a quantum channel χ, we have

$$(I_r \otimes \chi)(M) = \begin{bmatrix} \chi(A_{11}) & \chi(A_{12}) & \cdots & \chi(A_{1r}) \\ \chi(A_{21}) & \chi(A_{22}) & \cdots & \chi(A_{2r}) \\ \vdots & \vdots & \ddots & \vdots \\ \chi(A_{r1}) & \chi(A_{r2}) & \cdots & \chi(A_{rr}) \end{bmatrix}. \tag{490}$$

This leads to an alternative definition of $(I_r \otimes \chi)$: the input density matrix ρ is expressed as a block matrix of the form of Eq. (487) and then χ is applied to each block, resulting in a block matrix of the form of Eq. (490). Our *original* definition of $I_r \otimes \chi$ is in terms of the Kraus operators of the original channel, where the Kraus operators for the resulting channel on the combined system are of the form in Eq. (481). The two definitions of $(I_r \otimes \chi)$ are equivalent.

31.3 The partial transpose as an entanglement test

Recall from section 31.1 that the transpose $f_T : \mathbb{C}^{d \times d} \to \mathbb{C}^{d \times d}$, defined as $f_T(\rho) = \rho^T$ is a linear map that maps valid density matrices to valid density matrices. The *partial transpose* on a two-register system is the mapping $I_r \otimes f_T$, that applies the transpose to the second register.

Suppose that we have two qubit registers and we apply the transpose to the second

qubit. Applying $I_2 \otimes f_T$ to the Bell state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$, results in

$$(I_2 \otimes f_T)\left(\begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{bmatrix}\right) = \begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}. \tag{491}$$

What's remarkable about the resulting matrix is that *it is not a valid density matrix* (because it is not positive; it has a negative eigenvalue).

Exercise 31.2. Show that the eigenvalues of the matrix on the right side of Eq. (491) are $\frac{1}{2}$ (with multiplicity 3) and $-\frac{1}{2}$ (with multiplicity 1).

The fact that applying $I_2 \otimes f_T$ to the density matrix of the state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ does not yield a valid density matrix has these two interesting consequences.

Theorem 31.1. *The transpose f_T is not a quantum channel.*

Proof. If f_T were a valid quantum channel then $I_2 \otimes f_T$ would also be a valid quantum channel, so applying $I_2 \otimes f_T$ to the density matrix of the Bell state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ would result in a valid density matrix, which is not the case (by Eq. (491)). ■

Theorem 31.2. *The Bell state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ is not separable.*

Proof. It the Bell state were separable then its density matrix could be expressed as

$$\sum_{k=1}^{r} p_k \Big(\rho_k \otimes \sigma_k\Big), \tag{492}$$

where $(p_1, \ldots, p_r)$ is a probability vector, and $\rho_1, \ldots, \rho_r$, $\sigma_1, \ldots, \sigma_r$ are 2×2 density matrices. Then applying $I_2 \otimes f_T$ to this state would yield

$$\sum_{k=1}^{r} p_k \Big(\rho_k \otimes \sigma_k^T\Big), \tag{493}$$

which is a valid density matrix, contradicting Eq. (491). ■

Positive partial transpose test

For a general two-register quantum state ρ, if $(I_r \otimes f_T)(\rho)$ is not a valid density matrix then ρ cannot be a separable state, so ρ must be entangled.

What about the converse?

Theorem 31.3. *In the special case where ρ is a density matrix for two* qubit *registers, $(I_2 \otimes f_T)(\rho)$ is a valid density matrix if and only if ρ is separable.*

We do not prove Theorem 31.3 here.

In the general case, where the registers can have higher dimensions than 2, if $(I_r \otimes f_T)(\rho)$ is a valid density matrix then it can go either way: ρ could be separable or ρ could be entangled.

32 State transitions in the Stinespring form

In the previous section, I showed you how to express quantum measurements and quantum channels in terms of Kraus operators. In this section, I'm going to show you another form for expressing state transitions, called the *Stinespring form*.

32.1 Measurements in the Stinespring form

Imagine that the input state is a d-dimensional register. First, we append an m-dimensional ancilla register in some computational basis state, say $|0\rangle$. The combined system is md-dimensional. Next, we apply some $md \times md$ unitary operation U to the combined system. Finally, we measure one register in the computational basis, yielding a classical outcome k and a residual quantum state in the other register.

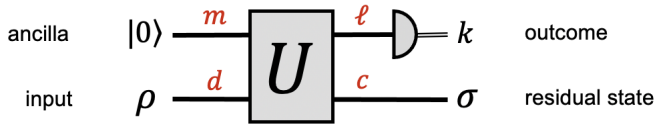

Figure 211: Quantum circuit for a measurement in the Stinespring form.

It's natural for the dimensions of the registers coming out of U to be the same as those of the registers going in ($\ell = m$ and $c = d$). But we allow for the dimensions of the outgoing registers to be different, as long as the total dimension is the same ($md = \ell c$). To get a feeling for this, consider the case where all the dimensions are powers of 2. In that case, we can assume that each register is a bunch of qubits.

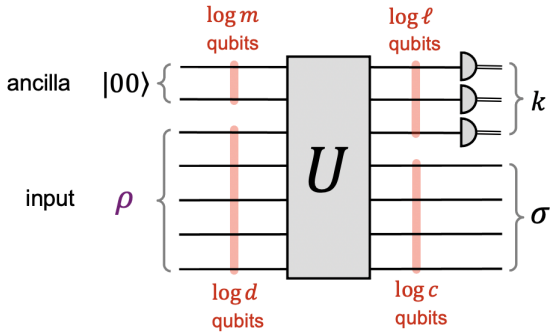

Figure 212: Quantum circuit on qubits for a measurement in the Stinespring form.

In this example, the input state is 5 qubits, whereas the residual outgoing state is 4 qubits. Also, the ancilla is 2 qubits, whereas the number of qubits that are measured

is 3. As long as the total number of qubits going into U and coming out of U is preserved, this makes perfect sense.

Also, note that some of the dimensions can be 1. A 1-dimensional register is essentially the same as no register. The only possible 1×1 density matrix is $I_1 = [1]$ and $I_1 \otimes \rho = \rho$. For example, for a measurement in an orthonormal basis specified by U, a very strict translation into the form of figure 212 is obtained by setting $m = c = 1$ and $\ell = d$ (where $U = I$ in the case of the computational basis).

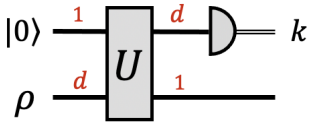

Figure 213: Measurement with respect to an orthonormal basis specified by U.

But figure 213 is pedantic, and we can freely omit the wires of dimension 1 (and omit any I gates). With this relaxation, we can denote a measurement with respect to an orthonormal basis in the Stinespring form as follows.

Figure 214: Measurement with respect to the computational basis and a basis specified by U.

Remember the "exotic measurements" in section 9.2 of *Part I*? It should be clear that those measurements are subsumed by these Stinespring measurements.

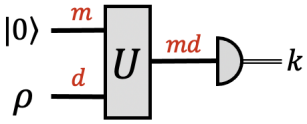

Figure 215: Exotic measurement in the Stinespring form.

32.2 Channels in the Stinespring form

As we noted in section 30.2, one way of thinking about a channel is as a measurement where we don't look at the classical part of the outcome. So we could define Stinespring channels that way. We'll do that, but we'll simplify things by noting that, if we're not going to see the classical outcome then, instead of performing the measurement, we can trace out that register, like this.

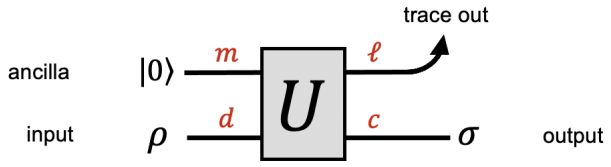

Figure 216: Quantum circuit for a channel in the Stinespring form.

Notice the circuit notation that I'm using here for tracing out a register: the imagery is supposed evoke that the register is tossed away.

Here are some of our most basic channels in the Stinespring form (loosely in the form of figure 216).

Figure 217: Unitary channel, add ancilla $|0\rangle\langle 0|$ channel, and partial trace Tr_1 channel.

All these channels are rather trivial examples. In the next subsections, we review some examples that are a little more interesting.

32.2.1 Decoherence of a qubit

The qubit decoherence channel was defined in the Kraus form in section 30.2.2. One way of thinking about this channel is that it produces the residual state when a qubit is measured in the computational basis (but where we don't see the classical part of the outcome). Here's a Stinespring circuit for the decoherence channel.

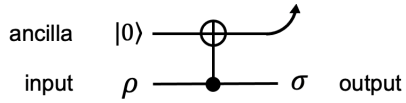

Figure 218: Decoherence of a qubit channel.

How does this work? Consider the case of a pure state $\alpha_0 |0\rangle + \alpha_1 |1\rangle$. The CNOT gate causes the state of the two qubits to become $\alpha_0 |00\rangle + \alpha_1 |11\rangle$, and then tracing out the first qubit yields the mixed state

$$\begin{bmatrix} |\alpha_0|^2 & 0 \\ 0 & |\alpha_1|^2 \end{bmatrix} = |\alpha_0|^2 |0\rangle\langle 0| + |\alpha_1|^2 |1\rangle\langle 1| , \tag{494}$$

which is consistent with the definition of this channel. So at least this works for the special case of pure states.

Exercise 32.1 (easy)**.** Show that the circuit in figure 218 implements the decoherence channel, as defined in section 30.2.2.

32.2.2 Reset channel

Here's a very simple channel that I'll call the *reset channel*. The input is a qubit and the output is a qubit in state $|0\rangle$ (regardless of what the input state is). Here's a very simple Stinespring circuit for it.

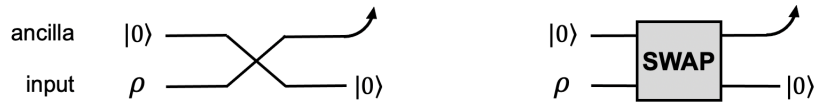

Figure 219: Circuit for the reset channel (with two different notations for the SWAP gate).

It's obvious that this circuit works: it traces out the input qubit and produces a qubit in state $|0\rangle$ as output.

Exercise 32.2. Express the reset circuit in the Kraus form (in terms of Kraus operators). (Hint: two Kraus operators suffice.)

Later in this section, we will see recipes for converting between the Stinespring form and the Kraus form, but there is a simple solution to the above which you might try to discover directly.

32.2.3 Depolarizing channel

The *depolarizing channel* was defined in the Kraus form in section 30.2.4. It is used as a natural model of noise, and we'll be seeing more of this channel when we get to the subject of quantum error-correcting codes in sections 35 and 36.

The channel is parameterized by $q \in [0, 1]$, and it maps an input qubit in state $\rho \in \mathbb{C}^{2\times 2}$ to an output qubit in state

$$(1-q)\rho + q \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}. \tag{495}$$

(This is slightly different scaling than for the parameter p in section 30.2.4, but is equivalent, by the conversion $q = \frac{4}{3}p$.)

An interpretation of this is that, with probability $1-q$, the state is left alone and with probability q the state is changed to the maximally mixed state $\begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$.

Can we represent this channel in Stinespring form? Here's one Stinespring circuit for this channel, where $R = \begin{bmatrix} \sqrt{1-q} & -\sqrt{q} \\ \sqrt{q} & \sqrt{1-q} \end{bmatrix}$.

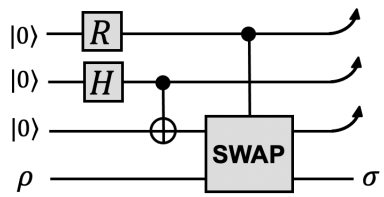

Figure 220: The depolarizing channel in the Stinespring form.

At first glance, this circuit may look complicated. But we can understand it by first looking at the two middle qubits. The H and CNOT gate are manufacturing a Bell state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$. Note that each qubit of the Bell state is in state $\begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$. Then the rest of the circuit applies

$$\begin{cases} \text{SWAP} & \text{with prob. } q \\ I & \text{with prob. } 1-q. \end{cases} \tag{496}$$

This can be seen by noting that the circuit is equivalent to the circuit in figure 221, where we are using the deferred measurement lemma (Lemma 8.1, from *Part I*) to

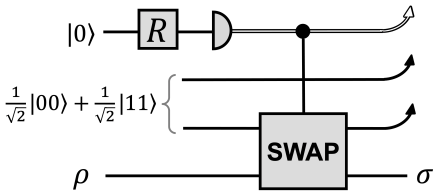

Figure 221: An equivalent circuit for the depolarizing channel.

measure the control qubit of the controlled-SWAP gate. This Stinespring form uses three ancilla qubits. Can this channel be implemented with fewer ancilla qubits?

A very easy way to reduce the ancilla to two qubits is to skip the Bell state and just initialize one ancilla qubit to the mixed state $\frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1|$. Technically, this isn't in the Stinespring form of figure 216, since that requires all ancilla qubits to be initialized to a pure state. But a construction allowing an ancilla to be initialized to a mixed state might nevertheless be useful in some contexts.

However, it turns out that there is a different Stinespring circuit for the depolarizing channel that uses only two ancilla qubits initialized to state $|00\rangle$. The construction is elegant and I leave it as an exercise.

Exercise 32.3. Give a Stinespring form for the depolarizing channel that uses only two ancilla qubits in initial state $|00\rangle$.

32.3 Equivalence of Kraus and Stinespring channels

Recall from Definition 30.1 that $A_0, A_1, \ldots, A_{\ell-1}$ is a sequence of Kraus operators if

$$\sum_{k=0}^{\ell-1} A_k^* A_k = I. \tag{497}$$

Associated with any sequence of Kraus operators, we have two transformations: a Kraus measurement and a Kraus channel. We're now going to prove two theorems.

Theorem 32.1 (Kraus to Stinespring)**.** *Any transformation in the Kraus form can be simulated in the Stinespring form.*

Theorem 32.2 (Stinespring to Kraus)**.** *Any transformation in the Stinespring form can be simulated in the Kraus form.*

32.3.1 Kraus to Stinespring

In this section we prove Theorem 32.1. For Kraus operators $A_0, A_1, \ldots, A_{\ell-1} \in \mathbb{C}^{c\times d}$, consider the block matrix

$$\begin{bmatrix} A_0 \\ A_1 \\ \vdots \\ A_{\ell-1} \end{bmatrix}, \tag{498}$$

which is an $\ell c \times d$ matrix. The columns of this matrix are orthonormal because

$$\begin{bmatrix} A_0^* & A_1^* & \cdots & A_{\ell-1}^* \end{bmatrix} \begin{bmatrix} A_0 \\ A_1 \\ \vdots \\ A_{\ell-1} \end{bmatrix} = \sum_{k=0}^{\ell-1} A_k^* A_k = I. \tag{499}$$

One consequence of this is that $d \leq \ell c$. Otherwise, the number of orthonormal vectors would have to exceed the dimension of the space in which they exist, which is impossible.

So we have ℓ Kraus operators that are $c \times d$ matrices and $d \leq \ell c$. To make the dimensions work out nicely, I'd like to assume that d divides ℓc. For this to hold, we might have to increase ℓ. It's straightforward to show that this can be done, where the new value of ℓ is less than double the original value of ℓ. If ℓ is increased we can add more Kraus operators that are zero matrices. Note that the larger set of matrices are still Kraus operators. So we can assume that d divides ℓc and set $m = \frac{\ell c}{d}$. Then we have $\ell c = md$.

Now, consider to the block matrix in Eq. (498) of Kraus operators again. Since its columns are orthonormal, we can extend this set of ℓc-dimensional column vectors to be an orthonormal basis of size ℓc. If we add these column vectors to the block matrix, we end up with a square unitary matrix. Call this $\ell c \times \ell c$ matrix U, which is of the form

$$U = \left[\begin{array}{c|c} \begin{matrix} A_0 \\ A_1 \\ \vdots \\ A_{\ell-1} \end{matrix} & W \end{array}\right]. \tag{500}$$

Now consider this circuit.

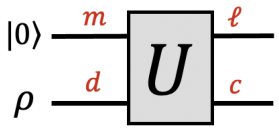

Figure 222: Stinespring circuit (omitting the final measurement/trace-out stage).

The input to the circuit consists of two registers: an m-dimensional ancilla and our d-dimensional input state. The circuit applies U to this. We can calculate the density

matrix of the output state as

$$U\Big(|0\rangle\langle 0| \otimes \rho\Big)U^* = \left[\begin{array}{c|c} A_0 & \\ A_1 & \\ \vdots & W \\ A_{\ell-1} & \end{array}\right] \begin{bmatrix} \rho & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \left[\begin{array}{c|c} A_0^* \quad A_1^* \quad \cdots \quad A_{\ell-1}^* & \\ \hline & W^* \end{array}\right] \tag{501}$$

$$= \left[\begin{array}{c|c} A_0\rho & \\ A_1\rho & \\ \vdots & 0 \\ A_{\ell-1}\rho & \end{array}\right] \left[\begin{array}{c|c} A_0^* \quad A_1^* \quad \cdots \quad A_{\ell-1}^* & \\ \hline & W^* \end{array}\right] \tag{502}$$

$$= \begin{bmatrix} A_0\rho A_0^* & A_0\rho A_1^* & \cdots & A\rho A_{\ell-1}^* \\ A_1\rho A_0^* & A_1\rho A_1^* & \cdots & A\rho A_{\ell-1}^* \\ \vdots & \vdots & \ddots & \vdots \\ A_{\ell-1}\rho A_0^* & A_{\ell-1}\rho A_1^* & \cdots & A_{\ell-1}\rho A_{\ell-1}^* \end{bmatrix}. \tag{503}$$

For the Stinespring channel, the final step is to trace out the first register. This partial trace is the sum of the $c \times c$ blocks along the diagonal, which is

$$\sum_{k=0}^{\ell-1} A_k \rho A_k^*. \tag{504}$$

For the Stinesrping measurement, the final step is to measure the first register in the computational basis. This measurement is defined in the Kraus form in section 30.1.1, and it is straightforward to deduce that the probability that the outcome of this measurement is k is $\mathrm{Tr}(A_k \rho A_k^*)$.

32.3.2 Stinepring to Kraus

Here I give a brief overview of the proof of Theorem 32.2. The Stinespring form consists of three stages. The first two are: adding an ancilla in state $|0\rangle$; and then applying a unitary operation U. The third stage is the partial trace for the Stinespring channel, and the measurement of the first register for the Stinespring measurement. Note that, in section 30, we have Kraus forms for each of these individual operations. We can compose these Kraus forms to obtain a Kraus channel from a Stinespring channel. We can compose them to obtain a Kraus measurement from a Stinespring measurement.

32.4 Unifying measurements and channels

I have been describing state transitions as if there's a clear dichotomy between measurements and channels. You either measure and get a classical outcome and a residual state or you apply a channel and get just a quantum state as outcome. In fact, there's a general notion that unifies these.

Let $f : \{0, 1, \ldots, \ell - 1\} \to T$ be some function. Suppose that we apply the Kraus measurement and then apply f to the classical outcome. So the classical outcome is $f(k)$, rather than k.

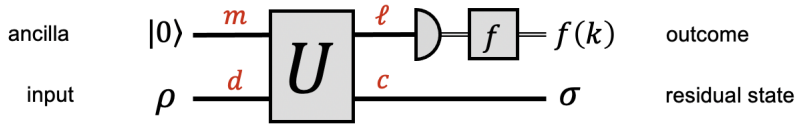

Figure 223: Generalized quantum transformation.

If f is a constant function, then seeing $f(k)$ provides us with no information about k. So that corresponds to the case of a channel. The other extreme case is where f is a bijection, for which knowing $f(k)$ provides full information about k. And there are in-between cases where f is not constant nor a bijection. In those cases, we receive *partial* information about k. The classical outcome $f(k)$ might narrow down the possible values of k, but without uniquely determining k.

33 Distance measures between states

This section is about distance measures between quantum states. We'll see various ways of quantifying how different two quantum states are, including the *fidelity* and the *trace distance*. I'll also show you the Holevo-Helstrom Theorem, which relates trace distance to the operational problem of distinguishing between a pair of states by a measurement.

Recall that we consider two quantum states to be *indistinguishable* if, for any measurement procedure, the probability distribution of outcomes is identical between the two states. For example, $|0\rangle$ and $-|0\rangle$ are indistinguishable. In section 28.2, we saw examples of different probabilistic mixtures that resulted in the same density matrix. In all such cases, the two states are indistinguishable; we don't even consider them to be different states.

Definition 33.1 (distinguishable states)**.** We say two states are *distinguishable* if they're not indistinguishable. In other words, if for some measurement procedure, the outcome probabilities are different for the two states.

For example, the $|0\rangle$ and the $|+\rangle$ state are distinguishable. Note that our definition of distinguishable does not require us to be able to perfectly tell the two states apart. Another example is $\frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1|$ and $|+\rangle\langle +|$.

Definition 33.2 (perfectly distinguishable states)**.** Define two states to be *perfectly distinguishable* if there is a measurement procedure that perfectly tells them apart.

For example, $|+\rangle$ and $|-\rangle$ are perfectly distinguishable.

We have three qualitative categories: indistinguishable, distinguishable, and perfectly distinguishable. Can we quantify how different two states are?

33.1 Operational distance measure

An operational way of quantifying the difference between two states is based on the guess-the-state game that we've seen several times.

For any two states, which in general can be mixed states, imagine the game where Alice flips a fair coin to decide which of the two states to set a quantum system to, and then she sends the quantum system to Bob. Bob knows what the two possible states are, but Alice does not tell him which one she chose. Bob's goal is to apply a measurement procedure measurement to the state that he received and to use the

classical outcome to guess which state it was. We can write the success probability of Bob's optimal measurement procedure as $(1+\delta)/2$, where $\delta \in [0,1]$.

The trivial strategy for Bob is to just guess a random bit (ignoring the system that he receives from Alice). That strategy succeeds with probability $\frac{1}{2}$. We can think of δ as how much better one can do than that baseline.

Another way of viewing δ is as the success probability minus the failure probability (which holds because $\delta = \frac{1+\delta}{2} - \frac{1-\delta}{2}$). It's sometimes useful to think of δ in that way.

For a given pair of quantum states, what's the best δ attainable? If the two states are indistinguishable then $\delta = 0$ is the best possible. At the other extreme, if the two states are perfectly distinguishable then the success probability can be 1, so $\delta = 1$.

An in-between case is when the two states are $|0\rangle$ and $|+\rangle$. These are not perfectly distinguishable, but the distinguishing probability can be as high as

$$\cos^2(\tfrac{\pi}{8}) = \frac{1+\cos(\frac{\pi}{4})}{2} = \frac{1+\frac{1}{\sqrt{2}}}{2}, \tag{505}$$

so $\delta = \frac{1}{\sqrt{2}} \approx 0.707$. Another in-between case is $\frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1|$ vs. $|+\rangle\langle +|$, for which the best possible distinguishing probability is $\frac{3}{4} = \left(1+\frac{1}{2}\right)/2$, so in that case $\delta = \frac{1}{2}$.

This is one natural way of quantifying the distance between two states, in terms of the highest distinguishing probability possible. A natural question is: given two density matrices ρ_0 and ρ_1, what is the highest distinguishing probability possible? In section 33.5, we'll see a systematic way of addressing such questions.

33.2 Geometric distance measures

Now, let's look at the distance between two quantum states form a different perspective: a geometric perspective.

33.2.1 Euclidean distance

If we have two d-dimensional pure states, $|\psi_0\rangle$ and $|\psi_1\rangle$ then it seems natural to take the Euclidean distance between their state vectors $\big||\psi_0\rangle - |\psi_1\rangle\big|_2$.

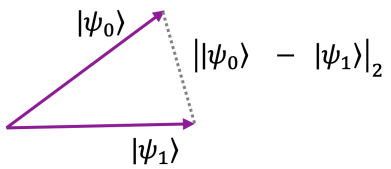

Figure 224: Euclidean distance between pure states $|\psi_0\rangle$ and $|\psi_1\rangle$.

If their Euclidean distance is small then it's intuitively natural to think of the states as "close". But, if their Euclidean distance is large, should we think of the states as "far apart"? Not necessarily, because of global phases. Note that $-|\psi\rangle$ is the unit vector farthest away from $|\psi\rangle$ (the Euclidean distance being 2), even though these are the same state.

Also, it is not so clear how this notion of Euclidean distance can be extended to mixed-states.

33.2.2 Fidelity

Another notion of distance is called the *fidelity*, which for pure states is the absolute value of the inner product between the state vectors $\big|\langle\psi_0|\psi_1\rangle\big|$.

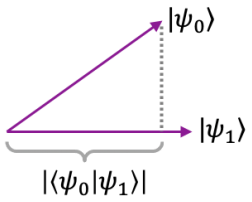

Figure 225: Fidelity between pure states $|\psi_0\rangle$ and $|\psi_1\rangle$.

This distance measure is calibrated in reverse in the sense that *large* fidelity means "close" and *small* fidelity means "far". Clearly, fidelity 1 means indistinguishable and fidelity 0 means perfectly distinguishable (since orthogonal states are perfectly distinguishable). Notice how the absolute value takes care of any distinctions between vectors due to global phases.

For the in-between values of fidelity, if the fidelity is close to 1 means that the states are "close".

What about the fidelity between mixed states? It turns out that there is a definition of fidelity for mixed states. If ρ_0 and ρ_1 are the density matrices of the two mixed states, then the fidelity is given by the formula

$$F(\rho_0, \rho_1) = \operatorname{Tr}\left(\sqrt{\sqrt{\rho_0}\,\rho_1\sqrt{\rho_0}}\right). \tag{506}$$

I'm not going to explain this formula, but I want you to see it. You cannot simplify this formula by cyclically permuting one of the $\sqrt{\rho_0}$ factors to the other side because of the square root within the trace. One reasonable property that this has is that, it agrees with the definition $\big|\langle\psi_0|\psi_1\rangle\big|$ for the very special case of pure states. This is easy to verify, which I'll leave as an exercise.

Exercise 33.1. Prove that, if $\rho_0 = |\psi_0\rangle\langle\psi_0|$ and $\rho_1 = |\psi_1\rangle\langle\psi_1|$ then it holds that $F(\rho_0, \rho_1) = \left|\langle\psi_0|\psi_1\rangle\right|$.

After looking at that expression for fidelity involving all those square roots of matrices, let's think about what it means to take the square root of a matrix. This is part of a more general *functional calculus* on square matrices.

33.3 Functional calculus for linear operators

Suppose that M is a normal matrix and $f : \mathbb{C} \to \mathbb{C}$. Then we can define f applied to the matrix M as follows. Since M is normal, we can diagonalize M in some orthonormal basis (the columns of some unitary U) as

$$M = U \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_d \end{bmatrix} U^*. \tag{507}$$

Define $f(M)$ as the matrix where f is applied to each eigenvalue, namely,

$$f(M) = U \begin{bmatrix} f(\lambda_1) & 0 & \cdots & 0 \\ 0 & f(\lambda_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f(\lambda_d) \end{bmatrix} U^*. \tag{508}$$

Square root of a positive matrix

Every $x \in \mathbb{C}$ has at least one square root, and if $x \neq 0$ then x has two square roots. However, if $x \in \mathbb{R}_+ = \{x \in \mathbb{R} : x \geq 0\}$ then x has a unique square root in $\mathbb{R}_+$. Whenever $M \geq 0$, there is a natural definition of $\sqrt{M}$ since then the eigenvalues of M are in $\mathbb{R}_+$ and have unique square roots in $\mathbb{R}_+$.

A word of caution: in general, for positive matrices L and M, it does *not* hold that $\sqrt{LM} = \sqrt{L}\sqrt{M}$. This is because L and M may not be simultaneously diagonalizable. They are each diagonalizable, but not necessarily with respect to the same orthonormal basis.

Von Neumann entropy of a positive matrix

Whenever $M \geq 0$, it makes sense to define $M \log M$ (which is related to the von Neumann Entropy of a quantum state ρ, defined as $\mathrm{Tr}\Big(\rho \log \rho\Big)$), that will come up in a later part of the course.

Absolute value of any matrix

We can also define the absolute value of a normal matrix M using the functional calculus, as the absolute values of all the eigenvalues

$$|M| = U \begin{bmatrix} |\lambda_1| & 0 & \cdots & 0 \\ 0 & |\lambda_2| & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & |\lambda_d| \end{bmatrix} U^*. \tag{509}$$

Notice that

$$|M| = \sqrt{M^* M}, \tag{510}$$

and this interesting because for *any* (not necessarily normal) matrix M, it holds that $M^*M \geq 0$. Therefore, we have a definition of the absolute value of a matrix that extends to *any* matrix (not necessarily diagonalizable).

33.4 Trace norm and trace distance

Now, I'd like to show you a very interesting distance measure between quantum states, called the trace distance. First, I'll show you the definition of the trace norm of a matrix, which is the trace of the absolute value of the matrix.

Definition 33.3 (trace norm)**.** For any $M \in \mathbb{C}^{d \times d}$, the *trace norm* of M is defined as

$$\|M\|_1 = \mathrm{Tr}\Big(|M|\Big) = \mathrm{Tr}\Big(\sqrt{M^* M}\Big). \tag{511}$$

The notation is as a norm with subscript 1 (sometimes this alternative notation is used, with the subscript "tr").

It's not too hard to show that, if M is normal then the trace norm of M is the 1-norm of the vector of eigenvalues of M. In other words, if the eigenvalues of M are $\lambda_1, \lambda_2, \ldots, \lambda_d$ then

$$\|M\|_1 = |\lambda_1| + |\lambda_2| + \cdots + |\lambda_d|. \tag{512}$$

Now we are ready to define the trace distance between two states. It's the trace norm of the difference between their density matrices.

Definition 33.4 (trace distance)**.** For any two d-dimensional states, whose density matrices are ρ_0 and ρ_1, the *trace distance* between them is

$$\|\rho_0 - \rho_1\|_1. \tag{513}$$

Of all the different matrix norms on which we could base a distance measure, what's so special about the trace norm? Why is this a meaningful measure of distance between states? The answer is given by the amazing Holevo-Helstrom Theorem.

33.5 The Holevo-Helstrom Theorem

Remember the δ that arose in our discussion of state distinguishability? We defined δ to be the advantage over random guessing in the guess-the-state game. In fact, that δ is exactly the trace norm multiplied by $\frac{1}{2}$. So the trace norm coincides with how well the states can be distinguished in the guess-the-state game!

Theorem 33.1 (Holevo-Helstrom Theorem)**.** *Let ρ_0 and ρ_1 be the density matrices of two d-dimensional states. If one of these two states is prepared by the flip of a fair coin and then the best distinguishing procedure succeeds with probability*

$$\frac{1 + \frac{1}{2}\|\rho_0 - \rho_1\|_1}{2}. \tag{514}$$

(If the trace distance had been defined as $\frac{1}{2}\|\rho_0 - \rho_1\|_1$ instead of $\|\rho_0 - \rho_1\|_1$ then the factor of $\frac{1}{2}$ would not apear in Eq. (514). But we're stuck with the standard definition.)

To prove the Holevo-Helstrom Theorem, we need to show:

- There is a measurement whose success probability is $\frac{1}{2} + \frac{1}{4}\|\rho_0 - \rho_1\|_1$.
- No measurement can perform better than $\frac{1}{2} + \frac{1}{4}\|\rho_0 - \rho_1\|_1$.

33.5.1 Attainability of success probability $\frac{1}{2} + \frac{1}{4}\|\rho_0 - \rho_1\|_1$

In this section, we prove the attainability part of Theorem 33.1. Namely, that there exists a measurement that attains success probability $\frac{1}{2} + \frac{1}{4}\|\rho_0 - \rho_1\|_1$, where ρ_0 and ρ_1 are the density matrices of the states that we wish to distinguish between.

Note that, since ρ_0 and ρ_1 are Hermitian, $\rho_0 - \rho_1$ is also Hermitian. But notice that, because of the minus sign, $\rho_0 - \rho_1$ need not be positive. In general, $\rho_0 - \rho_1$ has some negative eigenvalues.

Consider the two projectors, Π_0 and Π_1, defined as follows. Let Π_0 be the projector onto the space of all eigenvectors of $\rho_0 - \rho_1$ whose eigenvalues are ≥ 0. Let Π_1 be the projector onto the space of all eigenvectors of $\rho_0 - \rho_1$ whose eigenvalues are < 0. Then Π_0 and Π_1 are orthogonal projectors and $\Pi_0 + \Pi_1 = I$. Therefore Π_0 and Π_1 are the elements of a POVM measurement.

We'll show applying this measurement, and then guessing

$$\begin{cases} \text{“}\rho_0\text{”} & \text{when the measurement outcome is } \Pi_0 \\ \text{“}\rho_1\text{”} & \text{when the measurement outcome is } \Pi_1 \end{cases} \tag{515}$$

succeeds with probability $\frac{1}{2} + \frac{1}{4}\|\rho_0 - \rho_1\|_1$.

First note that

$$\Big(\Pi_0 - \Pi_1\Big)\Big(\rho_0 - \rho_1\Big) = \Big|\rho_0 - \rho_1\Big|. \tag{516}$$

To see why this is so, think of $\Pi_0 - \Pi_1$ and $\rho_0 - \rho_1$ in diagonal form (they are simultaneously diagonalizable). $\Pi_0 - \Pi_1$ has a $+1$ eigenvalue in all the positions where the corresponding eigenvalue of $\rho_0 - \rho_1$ is non-negative. $\Pi_0 - \Pi_1$ has a -1 eigenvalue in all the positions where the corresponding eigenvalue of $\rho_0 - \rho_1$ is negative. Therefore, $\Pi_0 - \Pi_1$ flips the sign of all the negative eigenvalues of $\rho_0 - \rho_1$, resulting in $\Big|\rho_0 - \rho_1\Big|$.

Note that Eq. (516) implies that

$$\mathrm{Tr}\Big((\Pi_0 - \Pi_1)(\rho_0 - \rho_1)\Big) = \|\rho_0 - \rho_1\|_1. \tag{517}$$

We can also expand

$$\begin{aligned} \mathrm{Tr}\Big((\Pi_0 - \Pi_1)(\rho_0 - \rho_1)\Big) &= \mathrm{Tr}(\Pi_0\rho_0) - \mathrm{Tr}\,\Pi_0\rho_1) - \mathrm{Tr}(\Pi_1\rho_0) + \mathrm{Tr}(\Pi_1\rho_1) && (518) \\ &= \Big(\mathrm{Tr}(\Pi_0\rho_0) + \mathrm{Tr}(\Pi_1\rho_1)\Big) - \Big(\mathrm{Tr}(\Pi_0\rho_1) + \mathrm{Tr}(\Pi_1\rho_0)\Big), && (519) \end{aligned}$$

where $\frac{1}{2}\Big(\mathrm{Tr}(\Pi_0\rho_0)+\mathrm{Tr}(\Pi_1\rho_1)\Big)$ is the success probability,[47] and $\frac{1}{2}\Big(\mathrm{Tr}(\Pi_0\rho_1)+\mathrm{Tr}(\Pi_1\rho_0)\Big)$ is the failure probability.[1] So the success probability minus the failure probability is equal to $\frac{1}{2}$ times the trace distance.

Note that our proof of this part is constructive. It actually gives us a recipe to determine the optimal measurement for distinguishing between two mixed states:

[47] Averaged over the random choice of the state.

consider the matrix that is one density matrix subtracted from the other density matrix; take the projector to the positive eigenspace of this matrix and the projector to the negative eigenspace of this matrix; that's the POVM measurement that solves the distinguishing problem with success probability $\frac{1}{2} + \frac{1}{4}\|\rho_0 - \rho_1\|_1$.

33.5.2 Optimality of success probability $\frac{1}{2} + \frac{1}{4}\|\rho_0 - \rho_1\|_1$

So far, we've proved that a particular measurement attains the proposed success probability. But how do we know that there isn't an even better measurement? In this section, we prove the optimality part of Theorem 33.1. Namely, that there does not exist a measurement whose success probability exceeds $\frac{1}{2} + \frac{1}{4}\|\rho_0 - \rho_1\|_1$.

Let E_0 and E_1 be the POVM elements of any 2-outcome POVM measurement for distinguishing between ρ_0 and ρ_1. We'll prove an upper bound on it's performance.

First, notice that E_0 and E_1 are simultaneously diagonalizable.[48] Why? Because $E_1 = I - E_0$. So if E_0 is diagonal is some coordinate system then so is E_1. Therefore, we can write

$$E_0 = U \begin{bmatrix} p_0 & 0 & \cdots & 0 \\ 0 & p_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p_{d-1} \end{bmatrix} U^* \quad \text{and} \quad E_1 = U \begin{bmatrix} q_0 & 0 & \cdots & 0 \\ 0 & q_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & q_{d-1} \end{bmatrix} U^*. \tag{520}$$

The eigenvalues of E_0 and E_1 are between 0 and 1 and corresponding eigenvalues sum to 1. It follows that the largest eigenvalue of $E_0 - E_1$ is at most 1.

Definition 33.5 (infinity norm)**.** For a normal matrix M, the *infinity norm*[49] $\|M\|_\infty$ is defined as the absolute value of the largest eigenvalue of M.

In this language, we have that $\|E_0 - E_1\|_\infty \leq 1$.

We will make use of the following lemma, which is an instance of Hölder's inequality for matrices.[50]

Lemma 33.1. *For any Hermitian L and M,* $\mathrm{Tr}(LM) \leq \|L\|_\infty \|M\|_1$.

[48] Please note that this only holds because it's a 2-outcome POVM measurement. For POVM measurements with 3 or more outcomes, the matrices might not be simultaneously diagonalizable.

[49] This is equivalent to the *spectral norm*, which is defined for all matrices.

[50] A proof of Lemma 33.1 can be found in: B. Baumgartner (2011). "An inequality for the trace of matrix products, using absolute values." arXiv:1106.6189

Now, we can bound the success probability minus the failure probability (averaged over inputs) as

$$\begin{aligned}\tfrac{1}{2}\mathrm{Tr}(E_0\rho_0) + \tfrac{1}{2}\mathrm{Tr}(E_1\rho_1) - \tfrac{1}{2}\mathrm{Tr}(E_0\rho_1) - \tfrac{1}{2}\mathrm{Tr}(E_1\rho_0) &= \tfrac{1}{2}\,\mathrm{Tr}((E_0 - E_1)(\rho_0 - \rho_1)) \quad (521)\\ &\leq \tfrac{1}{2}\|E_0 - E_1\|_\infty \|\rho_0 - \rho_1\|_1 \quad (522)\\ &\leq \tfrac{1}{2}\|\rho_0 - \rho_1\|_1. \quad (523)\end{aligned}$$

It follows that the success probability is at most

$$\frac{1 + \frac{1}{2}\|\rho_0 - \rho_1\|_1}{2} = \tfrac{1}{2} + \tfrac{1}{4}\|\rho_0 - \rho_1\|_1. \quad (524)$$

This proves optimality and we have now completed the proof of the Holevo-Helstrom Theorem.

33.6 Ulhmann's Theorem

Now, let's recall the previous strange-looking definition of fidelity between general mixed states that I showed you earlier—but which I didn't explain. Using our language of the trace norm, we can rewrite the expression for fidelity as

$$F(\rho_0, \rho_1) = \left\| \sqrt{\rho_0}\sqrt{\rho_1} \right\|_1. \quad (525)$$

For any two density matrices, we can take purifications of them and then take the inner product of these purifications. The result will depend on which purification we choose. The following theorem relates the fidelity between mixed states with inner products of their purifications.

Theorem 33.2 (Ulhmann's Theorem)**.** *For any two mixed states ρ_0 and ρ_1, the fidelity between them is the maximum $\langle\phi_0|\phi_1\rangle$ taken over all purifications $|\phi_0\rangle$ and $|\phi_1\rangle$.*

For further details, please see pages 410–411 in the book by Nielsen and Chuang.[51]

[51] M. A. Nielsen and I. L. Chuang (2000). *Quantum Computation and Quantum Information.* Cambridge University Press.

33.7 Fidelity vs. trace distance

We've discussed fidelity and trace distance, mostly the latter. Known relationships between them are

$$1 - F(\rho_0, \rho_1) \leq \|\rho_0 - \rho_1\|_1 \leq \sqrt{1 - F(\rho_0, \rho_1)^2}. \tag{526}$$

In particular, suppose that we have two mixed states with density matrices ρ_0 and ρ_1, and we want to show that their trace distance is small. One way to do this is to construct purifications of them whose inner products are close to 1. By Ulhmann's Theorem and the second inequality above, for any purifications $|\phi_0\rangle$ and $|\phi_1\rangle$ we have

$$\|\rho_0 - \rho_1\|_1 \leq \sqrt{1 - \langle\phi_0|\phi_1\rangle^2}. \tag{527}$$

34 Schmidt decomposition

In this section, I explain the *Schmidt decomposition*, which is a useful canonical form for pure bipartite states.

34.1 States that are equivalent up to local unitaries

Suppose that Alice and Bob each have a qubit, and their joint state is

$$\tfrac{1}{2}|00\rangle + \tfrac{1}{2}|01\rangle + \tfrac{1}{2}|10\rangle + \tfrac{1}{2}|11\rangle . \tag{528}$$

They can convert this state to

$$\tfrac{1}{2}|00\rangle - \tfrac{1}{2}|01\rangle - \tfrac{1}{2}|10\rangle + \tfrac{1}{2}|11\rangle \tag{529}$$

without any interaction or communication by each performing a Z gate locally to their qubit. There are many other states that they can create by performing local unitaries to their respective qubits. Question: can they convert the above state to

$$\tfrac{1}{2}|00\rangle + \tfrac{1}{2}|01\rangle + \tfrac{1}{2}|10\rangle - \tfrac{1}{2}|11\rangle \tag{530}$$

in this manner? The answer is no, because their starting state, Eq. (528), is the product state $|+\rangle|+\rangle$ and every state that this can be converted to by local unitaries is also a product state; whereas the latter state is $\frac{1}{\sqrt{2}}|0\rangle|+\rangle + \frac{1}{\sqrt{2}}|1\rangle|-\rangle$, which is not a product state (it is equivalent to $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$, by local unitaries).

Here is a less obvious example. I claim that

$$\tfrac{1}{2}|00\rangle + \tfrac{1}{2}|01\rangle + \tfrac{1}{\sqrt{2}}|10\rangle \tag{531}$$

is equivalent, by applying local unitaries, to

$$\alpha_0|00\rangle + \alpha_1|11\rangle \tag{532}$$

for amplitudes α_0, α_1.

Can you figure out what the local unitaries and α_0, α_1 are? This is equivalent to finding an orthonormal basis $|\mu_0\rangle, |\mu_1\rangle$ for the first qubit and an orthonormal basis $|\phi_0\rangle, |\phi_1\rangle$ for the second qubit and $\alpha_0, \alpha_1 \in \mathbb{C}$ such that

$$\tfrac{1}{2}|00\rangle + \tfrac{1}{2}|01\rangle + \tfrac{1}{\sqrt{2}}|10\rangle = \alpha_0|\mu_0\rangle|\phi_0\rangle + \alpha_1|\mu_1\rangle|\phi_1\rangle . \tag{533}$$

Note that, although it is easy to see that

$$\tfrac{1}{2}\left|00\right\rangle + \tfrac{1}{2}\left|01\right\rangle + \tfrac{1}{\sqrt{2}}\left|10\right\rangle = \tfrac{1}{\sqrt{2}}\left|0\right\rangle\left|+\right\rangle + \tfrac{1}{\sqrt{2}}\left|1\right\rangle\left|0\right\rangle, \tag{534}$$

this is not a useful form because $|+\rangle$ and $|0\rangle$ are not orthogonal. This (admittedly naive) approach assumed that the basis used for the first qubit is $|0\rangle, |1\rangle$.

It turns out that Eq. (533) is satisfied by setting:

$$\alpha_0 = \cos(\tfrac{\pi}{8}) \qquad \alpha_1 = \sin(\tfrac{\pi}{8}) \tag{535}$$

$$|\mu_0\rangle = |+\rangle \qquad |\mu_1\rangle = |-\rangle \tag{536}$$

$$|\phi_0\rangle = \cos(\tfrac{\pi}{8})|0\rangle + \sin(\tfrac{\pi}{8})|1\rangle \qquad |\phi_1\rangle = -\sin(\tfrac{\pi}{8})|0\rangle + \cos(\tfrac{\pi}{8})|1\rangle\,. \tag{537}$$

This solution to Eq. (533) can be verified by calculation (making use of the fact that $\cos^2(\frac{\pi}{8}) - \sin^2(\frac{\pi}{8}) = 2\cos(\frac{\pi}{8})\sin(\frac{\pi}{8}) = \frac{1}{\sqrt{2}}$). But how was this solution found?

The Schmidt decomposition and it's proof provides us with a systematic approach for addressing questions like these.

34.2 A tensor sum decomposition lemma

Here we state and prove Lemma 34.1, which is useful for the Schmidt decomposition.

The lemma may seem intuitively obvious, but we give a formal proof. The proof makes use of matrices of the form $I \otimes \langle\phi|$, which are complementary to the matrices of the form $I \otimes |\phi\rangle$ that we investigated in section 29.4.

Lemma 34.1 (tensor sum decomposition)**.** *Let $|\phi_0\rangle, |\phi_1\rangle, \ldots, |\phi_{d-1}\rangle \in \mathbb{C}^d$ be an orthonormal basis and $|\psi\rangle \in \mathbb{C}^c \otimes \mathbb{C}^d$. Then there exist $|\nu_0\rangle, |\nu_1\rangle, \ldots, |\nu_{d-1}\rangle \in \mathbb{C}^c$ such that*

$$|\psi\rangle = \sum_{k=0}^{d-1} |\nu_k\rangle \otimes |\phi_k\rangle\,. \tag{538}$$

The vectors $|\nu_0\rangle, |\nu_1\rangle, \ldots, |\nu_{d-1}\rangle$ need not be orthogonal. And they are not normalized; however, $\| |\nu_0\rangle \|^2 + \| |\nu_1\rangle \|^2 + \cdots + \| |\nu_{d-1}\rangle \|^2 = 1$ holds.

Proof of Lemma 34.1. For each $k \in \{0, 1, \ldots, d-1\}$, set

$$|\nu_k\rangle = \Big(I_c \otimes \langle\phi_k|\Big)\,|\psi\rangle\,. \tag{539}$$

Then

$$|\psi\rangle = \left(I_c \otimes \sum_{k=0}^{d-1} |\phi_k\rangle\langle\phi_k|\right)|\psi\rangle \qquad \text{since } \sum_{k=0}^{d-1} |\phi_k\rangle\langle\phi_k| = I_d \tag{540}$$

$$= \sum_{k=0}^{d-1} \left(I_c \otimes |\phi_k\rangle\right)\left(I_c \otimes \langle\phi_k|\right)|\psi\rangle \qquad \text{by Lemma 6.1} \tag{541}$$

$$= \sum_{k=0}^{d-1} \left(I_c \otimes |\phi_k\rangle\right)|\nu_k\rangle \qquad \text{by Eq. (539)} \tag{542}$$

$$= \sum_{k=0}^{d-1} |\nu_k\rangle \otimes |\phi_k\rangle\,. \qquad \text{by Lemma 29.1} \tag{543}$$

■

34.3 Statement and proof of the Schmidt decomposition

Theorem 34.1 (Schmidt decomposition)**.** *Let $|\psi\rangle \in \mathbb{C}^d \otimes \mathbb{C}^d$ be any pure state. Then there exists an orthonormal basis $|\mu_0\rangle, \ldots, |\mu_{d-1}\rangle$ (for the first register), and an orthonormal basis $|\phi_0\rangle, \ldots, |\phi_{d-1}\rangle$ (for the second register), and $\alpha_0, \ldots, \alpha_{d-1} \geq 0$ such that*

$$|\psi\rangle = \sum_{k=0}^{d-1} \alpha_k |\mu_k\rangle \otimes |\phi_k\rangle \tag{544}$$

and $|\phi_0\rangle, \ldots, |\phi_{d-1}\rangle$ are eigenvectors of $\rho = \mathrm{Tr}_1\Big(|\psi\rangle\langle\psi|\Big)$.

Proof. Let $\rho = \mathrm{Tr}_1\Big(|\psi\rangle\langle\psi|\Big)$ and let $|\phi_0\rangle, \ldots, |\phi_{d-1}\rangle$ be an orthonormal basis of eigenvectors of ρ. Therefore, there exist $p_0, \ldots, p_{d-1} \geq 0$ with $p_0 + \cdots + p_{d-1} = 1$ such that

$$\rho = \sum_{k=0}^{d-1} p_k |\phi_k\rangle\langle\phi_k|\,. \tag{545}$$

By Lemma 34.1 (about tensor sum decompositions), there exist $|\nu_0\rangle, \ldots, |\nu_{d-1}\rangle \in \mathbb{C}^d$ such that

$$|\psi\rangle = \sum_{k=0}^{d-1} |\nu_k\rangle \otimes |\phi_k\rangle\,. \tag{546}$$

Since $|\phi_0\rangle, \ldots, |\phi_{d-1}\rangle$ were specially chosen as eigenvectors of $\rho = \mathrm{Tr}_1\Big(|\psi\rangle\langle\psi|\Big)$, something remarkable happens: the vectors $|\nu_0\rangle, \ldots, |\nu_{d-1}\rangle$ are orthogonal. To see why, we begin by expanding $|\psi\rangle\langle\psi|$ as

$$|\psi\rangle\langle\psi| = \left(\sum_{k=0}^{d-1} |\nu_k\rangle \otimes |\phi_k\rangle\right)\left(\sum_{\ell=0}^{d-1} \langle\nu_\ell| \otimes \langle\phi_\ell|\right) = \sum_{k=0}^{d-1}\sum_{\ell=0}^{d-1} |\nu_k\rangle\langle\nu_\ell| \otimes |\phi_k\rangle\langle\phi_\ell| \,. \tag{547}$$

Tracing out the first register results in

$$\mathrm{Tr}_1\Big(|\psi\rangle\langle\psi|\Big) = \sum_{k=0}^{d-1}\sum_{\ell=0}^{d-1} \mathrm{Tr}_1\Big(|\nu_k\rangle\langle\nu_\ell| \otimes |\phi_k\rangle\langle\phi_\ell|\Big) = \sum_{k=0}^{d-1}\sum_{\ell=0}^{d-1} \langle\nu_\ell|\nu_k\rangle\, |\phi_k\rangle\langle\phi_\ell| \tag{548}$$

(where the last equation is from $\mathrm{Tr}_1(A \otimes B) = \mathrm{Tr}(A)B$, from Eq. (427)). Combining this with Eq. (545), we have

$$\sum_{k=0}^{d-1}\sum_{\ell=0}^{d-1} \langle\nu_\ell|\nu_k\rangle\, |\phi_k\rangle\langle\phi_\ell| = \sum_{k=0}^{d-1} p_k\, |\phi_k\rangle\langle\phi_k| \,, \tag{549}$$

which implies that

$$\langle\nu_\ell|\nu_k\rangle = \begin{cases} p_k & \text{if } \ell = k \\ 0 & \text{if } \ell \neq k. \end{cases} \tag{550}$$

It follows that $|\nu_0\rangle, \ldots, |\nu_{d-1}\rangle$ are orthogonal (some of these vectors may be zero).

It is now simple to convert $|\nu_0\rangle, \ldots, |\nu_{d-1}\rangle$ to orthonormal vectors $|\mu_0\rangle, \ldots, |\mu_{d-1}\rangle$ such that $|\nu_k\rangle = \sqrt{p_k}\,|\mu_k\rangle$, for all $k \in \{0, \ldots, d-1\}$. For each k, if $|\nu_k\rangle \neq 0$ then set

$$|\mu_k\rangle = \frac{|\nu_k\rangle}{\||\nu_k\rangle\|} = \frac{|\nu_k\rangle}{\sqrt{p_k}}. \tag{551}$$

For the values of k for which $|\nu_k\rangle = 0$, set those $|\mu_k\rangle$ to any orthonormal basis for the orthogonal complement of $\mathrm{span}\{|\nu_0\rangle, \ldots, |\nu_{d-1}\rangle\}$, and set those $p_k = 0$. The resulting $|\mu_0\rangle, \ldots, |\mu_{d-1}\rangle$ is an orthonormal basis for $\mathbb{C}^d$ and $|\nu_k\rangle = \sqrt{p_k}\,|\mu_k\rangle$, for all $k \in \{0, \ldots, d-1\}$.

Setting $\alpha_k = \sqrt{p_k}$, for each k, we have our Schmidt decomposition

$$|\psi\rangle = \sum_{k=0}^{d-1} \alpha_k\, |\mu_k\rangle \otimes |\phi_k\rangle \,. \tag{552}$$

■

The amplitudes $\alpha_0, \ldots, \alpha_{d-1}$ are called the *Schmidt coefficients* of the state, and their standard ordering is $\alpha_0 \geq \alpha_1 \geq \cdots \geq \alpha_{d-1} \geq 0$. It is common to omit zero Schmidt coefficients from the sequence. The number of non-zero Schmidt coefficients is called the *Schmidt rank* of the state.

The Schmidt decomposition can be generalized to states $|\psi\rangle \in \mathbb{C}^{d_1} \otimes \mathbb{C}^{d_2}$, where the dimensions of the two registers are not the same. The resulting decomposition will be in terms of $d_{\min} = \min(d_1, d_2)$ orthonormal states for each register and $d_{\min}$ Schmidt coefficients. This is a simple adaptation of the above proof.

Incidentally, it is possible to deduce the Schmidt decomposition from the Singular Value Decomposition Theorem, though we don't show this here.

A word of caution: there is no Schmidt decomposition for general *tripartite* states. There exist states $|\psi\rangle \in \mathbb{C}^d \otimes \mathbb{C}^d \otimes \mathbb{C}^d$ that *cannot* be expressed as

$$|\psi\rangle = \sum_{k=0}^{d-1} \alpha_k |\eta_k\rangle \otimes |\mu_k\rangle \otimes |\phi_k\rangle . \tag{553}$$

for orthonormal bases $|\eta_0\rangle, \ldots, |\eta_{d-1}\rangle$, $|\mu_0\rangle, \ldots, |\mu_{d-1}\rangle$, $|\phi_0\rangle, \ldots, |\phi_{d-1}\rangle$ and coefficients $\alpha_0, \ldots, \alpha_{d-1}$. For example, $\frac{1}{\sqrt{3}}|100\rangle + \frac{1}{\sqrt{3}}|010\rangle + \frac{1}{\sqrt{3}}|001\rangle$ is a tripartite state with no tripartite Schmidt decomposition.

34.4 Applications of the Schmidt decomposition

One immediate application of the Schmidt decomposition is to be able to determine whether or or not, for two pure bipartite state $|\psi_1\rangle, |\psi_2\rangle \in \mathbb{C}^d \otimes \mathbb{C}^d$, one state can be transformed into the other state by *local* unitary operations. They can be transformed by local unitaries if and only if they have the same Schmidt coefficients, counting multiplicity, and in any order. Note in particular that applying local unitaries does not change the Schmidt coefficients, since

$$\Big(U \otimes V\Big) |\psi\rangle = \sum_{k=0}^{d-1} \alpha_k \Big(U|\mu_k\rangle\Big) \otimes \Big(V|\phi_k\rangle\Big) \tag{554}$$

$$= \sum_{k=0}^{d-1} \alpha_k |\mu'_k\rangle \otimes |\phi'_k\rangle , \tag{555}$$

where $|\phi'_k\rangle = U|\phi_k\rangle$ and $|\mu'_k\rangle = V|\mu_k\rangle$.

So, for pure states, the Schmidt coefficients are a characterization of entanglement. They represent properties of the state that do not change when the local coordinate

systems are changed. One way of quantifying the "amount of entanglement" of a pure bipartite state is in terms of its *Schmidt rank*, which is defined as the number of non-zero Schmidt coefficients. A more continuous measure of "amount of entanglement" is the *entanglement entropy*, which is defined as the Shannon entropy of the squares of the Schmidt coefficients.

Another application of the Schmidt decomposition is the Schrödinger-Hughston-Jozsa-Wootters Theorem (a.k.a. SHJW Theorem), which can be described as follows. Let $\rho \in \mathbb{C}^{d\times d}$ be the density matrix of some mixed state and let $|\psi_1\rangle, |\psi_2\rangle \in \mathbb{C}^d \otimes \mathbb{C}^d$ be any two bipartite states that are both purifications of ρ in the sense that $\mathrm{Tr}_1\Big(|\psi_1\rangle\langle\psi_1|\Big) = \mathrm{Tr}_1\Big(|\psi_2\rangle\langle\psi_2|\Big) = \rho$. Suppose that Alice is in possession of the first d-dimensional register and Bob is in possession of the second d-dimensional register of the state $|\psi_1\rangle$. The SHJW Theorem states that *Alice alone* can change the state from $|\psi_1\rangle$ to $|\psi_2\rangle$. In other words, there exists a unitary $U \in \mathbb{C}^{d\times d}$ such that

$$\Big(U \otimes I\Big)\,|\psi_1\rangle = |\psi_2\rangle\,. \tag{556}$$

35 Simple quantum error-correcting codes

In this section, we begin the subject of quantum error-correcting codes, which can protect quantum states from noise. We'll first briefly review some results about error-correcting codes for *classical* information. Then we'll consider *quantum* error-correcting codes and I'll explain Shor's nine-qubit quantum error-correcting code.

Let's start by discussing what noise is. Broadly speaking, noise is when information gets disturbed. Imagine that Alice wants to send some bits to Bob, but their communication channel is flawed.

Figure 226: Alice sending Bob a bit over a noisy communication channel.

Suppose that, for each bit $b \in \{0,1\}$ that Alice sends via the channel, what Bob receives is

$$\begin{cases} b & \text{with prob. } 1-\epsilon \\ \neg b & \text{with prob. } \epsilon, \end{cases} \tag{557}$$

for some parameter $\epsilon \in [0, \frac{1}{2}]$. So each bit gets flipped with probability ϵ. This mapping from states of bits to states of bits is called a *binary symmetric channel*, and we refer to it as BSC (or as BSC_ϵ, to specify the parameter ϵ).

This channel can be viewed as a classical analogue of the depolarizing channel. Recall that the depolarizing channel takes a qubit as input and produces as output a probabilistic mixture of that state and the maximally mixed state. The binary symmetric channel outputs a probabilistic mixture of the input bit and a classical maximally mixed state (which is a uniformly distributed random bit). If we identify bit 0 with the probability vector $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and bit 1 with $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ then the binary symmetric channel BSC_ϵ is a linear mapping such that

$$\mathsf{BSC}_\epsilon \begin{bmatrix} 1 \\ 0 \end{bmatrix} = (1-2\epsilon) \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 2\epsilon \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} \tag{558}$$

$$\mathsf{BSC}_\epsilon \begin{bmatrix} 0 \\ 1 \end{bmatrix} = (1-2\epsilon) \begin{bmatrix} 0 \\ 1 \end{bmatrix} + 2\epsilon \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix}. \tag{559}$$

Now suppose that Alice wants to communicate a bit $b \in \{0, 1\}$ to Bob and their communication channel is a binary symmetric channel BSC_ϵ. Can Alice and Bob reduce the noise level to a smaller ϵ? The obvious way is for Alice and Bob to get a better communication hardware, with a smaller parameter ϵ. But Alice and Bob have an alternative to investing in better hardware: they can use an error-correcting code.

35.1 Classical 3-bit repetition code

Perhaps the simplest error-correcting code is the 3-bit repetition code, which works as follows. To transmit bit $b \in \{0, 1\}$, Alice encodes b into three copies of b. Then Alice sends each of these three bits through the channel. Then Bob takes the majority value of the three bits that he receives (which might not all be the same, because some bits might get flipped by the channel).

How well does this perform? The system succeeds if no more than one bit is flipped (because that doesn't change the majority) and it fails if two or more bits are flipped. Assume that the channel behaves independently for each bit that passes through it.

Then the failure probability can be calculated as follows. There are three ways that two bits can be flipped, each occurring with probability $\epsilon^2(1 - \epsilon)$. And there is one way that all three bits can flip, occurring with probability ϵ^3. So the failure probability is

$$3\epsilon^2(1 - \epsilon) + \epsilon^3 = 3\epsilon^2 - 2\epsilon^3. \tag{560}$$

Here's a plot of the failure probability resulting from this scheme as a function of the original failure probability of the channel.

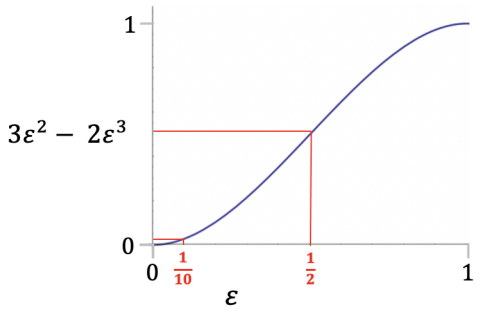

Figure 227: Failure probability of the 3-bit repetition code as a function of ϵ.

If $\epsilon = \frac{1}{2}$ then there is no improvement: the success probability remains at $\frac{1}{2}$. But this should not be surprising, because, if $\epsilon = \frac{1}{2}$ then the channel sends no information: it just outputs a random bit uncorrelated with the bit that Alice is sending. But when ϵ is smaller there is an advantage. For example, when $\epsilon = \frac{1}{10}$ the failure probability from using the code is around $\frac{1}{35}$. And the smaller ϵ is the more pronounced the error reduction is. If $\epsilon = \frac{1}{1000}$ then the failure probability from using the codes is around $\frac{1}{300000}$ (a three-hundred-fold decrease).

What price are we paying for this improvement? The main cost is that that three bits have to be sent instead of one.

Definition 35.1 (rate of a code)**.** The *rate* of a code is the inverse of the expansion in message length due to the encoding.

The rate of this code is $\frac{1}{3}$. Each bit of the encoding conveys $\frac{1}{3}$ of a bit of the data to be transmitted.

If the data is a long string of bits then there will be errors, but fewer errors using the code. With no code, the expected fraction of errors is ϵ. With the code, the expected fraction of errors is $3\epsilon^2 - 2\epsilon^3$. Suppose that Alice wants to send a Gigabyte to Bob (that's around 8.5 billion bits) and their communication channel has noise parameter $\epsilon = \frac{1}{100}$. Without the code, the fraction of errors will be around 85 million. With the code, the fraction of errors will be about 24 thousand. That's significantly fewer errors. It is achieved at the cost of sending three Gigabytes instead of one.

But suppose the we don't want *any* errors. Can this be achieved? One approach is to use a larger repetition code than 3-bits. That reduces the error probability for each bit. But this also reduces the rate—so, in the above example, many more Gigabytes would have to be sent. But there are much better error-correcting codes than repetition codes.

35.2 The existence of good classical codes in brief

An error-correcting code need not separately encode each bit. Rather, each block of n bits (a message) can be encoded into a block of m bits (a codeword). The rate of such a code is $\frac{n}{m}$. Note that a small rate means a large message expansion (an inefficiency); whereas, a rate close to 1 means a small message expansion.

A fundamental result about the existence of good classical error-correcting codes can be informally stated as:

> For a binary symmetric channel with any error parameter $\epsilon < \frac{1}{2}$, the success probability for encodings of long strings can be made arbitrarily close to 1, *while maintaining a constant rate*.

So, in fact, Alice doesn't have to send many Gigabytes in order to get every single bit through to Bob correctly.

I'm going to state the result about good error-correcting codes more precisely. Please note that I'm not going to give the details of the construction or the analysis. The theory of error-correcting codes is a large field of study, that could easily take a course of its own course to explain.

In order to state the result, we need some basic definitions for block codes. Such a code consists of an *encoding* function $E : \{0,1\}^n \to \{0,1\}^m$ and a *decoding* function $D : \{0,1\}^m \to \{0,1\}^n$. The rate of such a code is $\frac{n}{m}$.

Assuming that the communication channel is a binary symmetric channel with error parameter ϵ, the error probability of a code of the above form is defined as the maximum, for all $a_1 a_2 \dots a_n \in \{0,1\}^n$, of

$$\Pr\Big[D(\mathsf{BSC}_\epsilon(E(a_1 a_2 \dots a_n))) \Big] \neq a_1 a_2 \dots a_n. \tag{561}$$

Just to be clear: success means *all* the bits are successfully received at the other end; that there are no errors.

Definition 35.2 (Shannon entropy)**.** The *Shannon entropy* of a probability vector $(p_1, p_2, \dots, p_d)$ is defined as

$$H(p_1, p_2, \dots, p_d) = -\sum_{k=1}^{d} p_k \log p_k \tag{562}$$

(the logarithm is with respect to base 2 and we make the convention that $0 \log(0) = 0$).

Define $R : [0, \frac{1}{2}] \to [0,1]$ as $R(\epsilon) = 1 - H(\epsilon, 1-\epsilon)$. Here is a plot of this function.

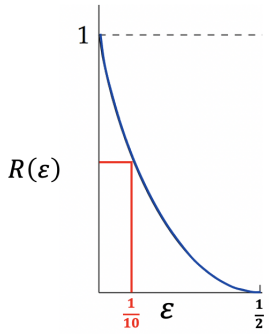

Figure 228: The rate function $R(\epsilon)$.

I'm now going to state the result about good multi-bit error-correcting codes. Let the noise level be any $\epsilon < 1/2$. Think of that as a property of the communication channel that you're stuck with using.

Then you can select any rate r, as long as $r < R(\epsilon)$. And you can select an arbitrarily small $\delta > 0$, which is your desired error probability bound. Then there exists an error-correcting code $E : \{0,1\}^n \to \{0,1\}^m$, $D : \{0,1\}^m \to \{0,1\}^n$ with rate $\frac{n}{m} > r$ and whose failure probability is less than δ.

There are additional considerations, that I'd like to mention, even though I won't go in the details:

- One is the block-length n. The smaller δ is, the larger n has to be.
- Another is the computational cost of computing E and D. The bottom line is that there are codes for which this can be done efficiently.

OK, so that was a very brief overview of error-correcting codes for classical information. The question is what happens with quantum error-correcting codes.

35.3 Shor's 9-qubit quantum error-correcting code

We started with a simple classical error-correcting code, the 3-bit repetition code. So we might be tempted to start with a quantum repetition code, where a qubit is encoded as three copies of itself.

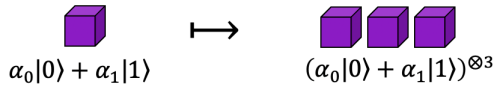

Figure 229: A naïve first attempt at a 3-qubit quantum repetition code.

Of course, this fails for multiple reasons, starting with the fact that a general quantum state cannot be copied (the no-cloning theorem).

It's easy to copy classical information, and all error-correcting codes for classical information are based on some sort of redundancy. But the no-cloning theorem kind of suggests that redundancy for quantum information might not be possible. That was the thinking shortly after Shor's algorithms for factoring and discrete log came out. But the underlying intuition that no-cloning implies no-redundancy was wrong.

I'm going to show you the first quantum error-correcting code, due to Shor.[52] It's a 9-qubit code that is constructed by combining two 3-qubit codes that protect against very limited error types.

[52] P. W. Shor (1995). "Scheme for reducing decoherence in quantum computer memory." *Phys. Rev. A*, **52**, R2493(R).

35.3.1 3-qubit code that protects against one X error

Let's start with a simple 3-qubit code that protects against a very restricted set of errors. Suppose the only error possible is a Pauli X, a bit flip. Then these encoding and decoding circuits protect against up to one X-error.

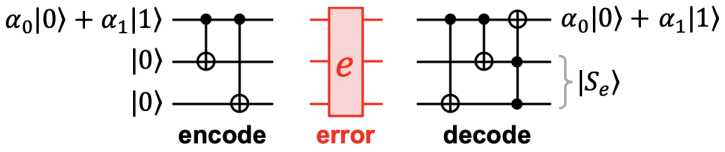

Figure 230: 3-qubit code that protects against one X error.

The *encoding* circuit takes a qubit as input and produces three qubits as output. Then the encoded data is affected by an error e, where e can be any one of these four unitary operations:

$I \otimes I \otimes I$	$X \otimes I \otimes I$	$I \otimes X \otimes I$	$I \otimes I \otimes X$
(no error)	(flip 1st qubit)	(flip 2nd qubit)	(flip 3rd qubit)

(563)

Then the three qubits are input to the *decoding* circuit.

It turns out that, in all four cases of e, the data is correctly recovered. The final state of the first qubit is the same as its initial state. It's a straightforward exercise to verify these, but I recommend that you work through some of these cases to convince yourself.

If you work out the final states, you will see that the second and third qubits end up in a state that depends on what the error operation e is: $|00\rangle$ (for $e = I \otimes I \otimes I$), $|11\rangle$ (for $e = X \otimes I \otimes I$), $|10\rangle$ (for $e = I \otimes X \otimes I$), and $|01\rangle$ (for $e = I \otimes I \otimes X$). We'll refer to these two qubits as the *syndrome of the error*, denoted as $|s_e\rangle$. So, not only does the encoding/decoding protect the data against the errors, but the decoding process also reveals what the error e that was corrected was.

What about other errors? Specifically, what happens if there is a Z-error on one of the three encoded qubits? A Z-error is not corrected; instead it's passed through. By this, I mean that if the data is in state $\alpha_0 |0\rangle + \alpha_1 |1\rangle$ then applying a Z-error to one of the three encoded qubits causes the output of the decoding circuit to be $\alpha_0 |0\rangle - \alpha_1 |1\rangle$, which is the same as applying Z directly to the original data. So this encoding/decoding does *not* protect against a Z-error; it passes Z errors through.

35.3.2 3-qubit code that protects against one Z error

Now, here's another 3-qubit code which protects against up to one Z-error.

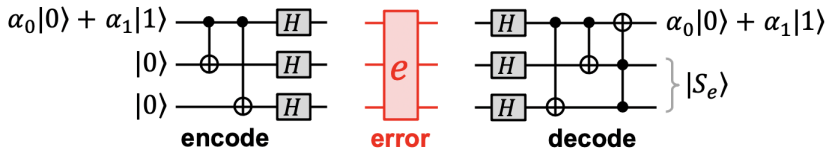

Figure 231: 3-qubit code that protects against one Z error.

The encoding/decoding is like the code in figure 230 for protecting against X-errors, but with a layer of Hadamard gates added to the end of the encoding and the beginning of the decoding. This is essentially a re-purposing of our code for X-errors into a code for Z-errors. Think of a Z-error and the H gates. Since $HZH = X$ and $HH = I$, any Z-errors effectively become X-errors and then are handled as the previous encoding/decoding in figure 230.

35.3.3 9-qubit code that protects against one Pauli error

By combining the 3-qubit codes from figures 230 and 231, we can obtain a 9-qubit code that protects against any Pauli error (I, X, Y, or Z) in any one of the nine qubit positions. The encoding/decoding circuits are the following.

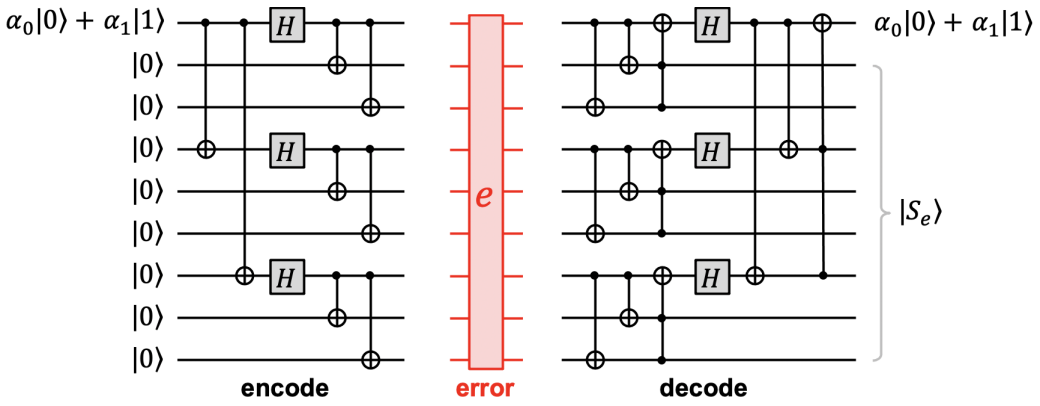

Figure 232: Shor's 9-qubit code.

A nice way of understanding how this code works is in terms of it's *inner part* (the last two layers of gates in the encoding part and first three layers of gates in the decoding part) and an *outer part* (the other gates). The inner part consists of three

blocks, where each block is a copy of our code for X-errors from figure 230. And the outer part is our code for Z-errors from figure 231.

What happens if there's an X-error? It's corrected by the inner part. What happens if there's a Z-error? A Z-error is passed through by the inner part and then corrected by the outer part. What happens if there's a Y-error? Since $Y = iXZ$ (and the phase i makes no difference in this context), this can be viewed as a Z-error and an X-error. The inner part corrects the X-error and the outer part corrects the Z-error.

So if the error e is any Pauli error in one qubit position then it is corrected by this code; the data is recovered in the final state of the first qubit. There are 28 ($= 3 \times 9 + 1$) different one-qubit Pauli errors e (including $I^{\otimes 9}$) that this code protects against. The final state of eight qubits after the first qubit is the *error syndrome* $|s_e\rangle$, and contains information about which error was corrected.[53]

In fact, this code corrects against an *arbitrary* error in any one qubit position and it is also effective for communicating through a channel with depolarizing noise.

35.4 Quantum error models

Let's step back and consider some of the error models that the Shor code protects against. There are several error models, but let's focus on two that are the most closely related to our discussion.

Worst-case unitary noise

By *worst-case unitary noise* noise, we mean that there is a set of possible unitary errors, and the error can be any one of them. For example, the Shor code is resilient against an arbitrary one-qubit Pauli error on a 9-qubit encoding.

Figure 233: Arbitrary unitary error on one single qubit.

In fact, if a code is resilient against any 1-qubit Pauli error then it is resilient against *any* 1-qubit unitary operation U. To see why this is so, note that any 2×2 unitary U can be expressed as

$$U = \eta_0 I + \eta_x X + \eta_y Y + \eta_z Z, \tag{564}$$

[53] A curiosity is that some of the 28 possible one-qubit Pauli errors have the error syndrome.

for some $\eta_0, \eta_x, \eta_y, \eta_z \in \mathbb{C}$. Then it's a straightforeward exercise to deduce from figure 232 that, if the error is set to $e = I^{\otimes j-1} \otimes U \otimes I^{\otimes 8-j}$ (that is, applying U to the j-th qubit), then the output will state is

$$\Big(\alpha_0 |0\rangle + \alpha_1 |1\rangle\Big) \otimes \Big(\eta_0 |s_0\rangle + \eta_x |s_{x,j}\rangle + \eta_y |s_{y,j}\rangle + \eta_z |s_{z,j}\rangle\Big), \tag{565}$$

where $|s_0\rangle$, $|s_{x,j}\rangle$, $|s_{y,j}\rangle$, $|s_{z,j}\rangle$ are the respective error syndromes of I, X, Y, Z in position j.

More generally, there exist other codes, with m-qubit encodings which have the property that, for some threshold k, the error can be an arbitrary unitray operation acting on any subset of the qubits of size k.

Depolarizing noise

Our discussion of classical error-correcting codes was based on the binary symmetric channel, which flips any bit passing through it with probability ϵ. A quantum analogue of the binary symmetric channel is the depolarizing channel, which can be defined as the mixed unitary channel of the form

$$\begin{cases} I & \text{with prob. } 1-\epsilon & \text{(no error)} \\ X & \text{with prob. } \epsilon/3 & \text{(bit flip)} \\ Y & \text{with prob. } \epsilon/3 & \text{(bit+phase flip)} \\ Z & \text{with prob. } \epsilon/3 & \text{(phase flip)} \end{cases} \tag{566}$$

for a parameter $\epsilon \leq \frac{3}{4}$. This is like the binary symmetric channel, but allowing for the fact that a qubit can be flipped in more than one way (bit-flip, phase-flip, or both).

Exercise 35.1. Show that the definition of the depolarizing channel in Eq. (566) is equivalent to our earlier definition of the depolarizing channel, as mapping each state ρ to a convex combination of that state and the maximally mixed state, namely

$$p\rho + (1-p)\Big(\tfrac{1}{2} |0\rangle\langle 0| + \tfrac{1}{2} |1\rangle\langle 1|\Big) \qquad \text{(for some } p\text{)}. \tag{567}$$

In the depolarizing noise model, we assume that every qubit of the encoded state is independently affected by a depolarizing channel. So all of the qubits incur an error; however, there is a bound ϵ on the severity of that error.

Figure 234: Depolarizing error on every qubit.

How does the Shor code perform in this model? For the depolarizing channel with parameter ϵ, let's think of ϵ as the *effective error probability*, since the qubit passes through the channel is undisturbed with probability $1-\epsilon$. If a qubit is encoded by the Shor code and each of the nine qubits of the encoding incurs a depolarizing error with parameter ϵ then we can analyze the effective error probability resulting from the encoding/decoding process. It turns out that this effective error probability is upper bounded $c\epsilon^2$, for some constant[54] c. So, if ϵ is small enough to begin with, then the reduction due to squaring ϵ is more than the effect of multiplying by c, so the effective error is decreased. The cost is that Alice has to send nine qubits to convey just one qubit. So the rate of this code is $\frac{1}{9}$.

Are there better codes than this? And are there good multi-qubit quantum error-correcting codes? This will be discussed in section 36.

35.5 Redundancy vs. cloning

The Shor code encodes one qubit in state $\alpha_0 |0\rangle + \alpha_1 |1\rangle$ into nine qubits, in state $\alpha_0 |0\rangle_L + \alpha_1 |1\rangle_L$ where

$$|0\rangle_L = \left(\tfrac{1}{\sqrt{2}} |000\rangle + \tfrac{1}{\sqrt{2}} |111\rangle\right)^{\otimes 3} \tag{568}$$

$$|1\rangle_L = \left(\tfrac{1}{\sqrt{2}} |000\rangle - \tfrac{1}{\sqrt{2}} |111\rangle\right)^{\otimes 3}. \tag{569}$$

Sometimes, $|0\rangle_L$ and $|1\rangle_L$ are referred to as the *logical zero* and *logical one* of the code.

The encoded state has the property that, if any error is inflicted on any one of its qubits then the data can still be recovered. To be clear, note that the recovery procedure is not provided with any information about *which* qubit has been affected by an error.

Which of the nine qubits contains the data? The answer is that no individual qubit contains any information about the qubit. The information is somehow "spread out" among the nine qubits.

A natural question is whether there is a more efficient code that requires fewer than nine qubits and protects against any one-qubit error. This question will be answered in section 36.

I'd like to briefly mention a different type of error, called an *erasure error*. For this type of error, the positions of the affected qubits are known. One way of viewing this

[54] I don't know the exact constant, but it is less than 36.

is that some of the qubits are "lost," but we know the positions of the lost qubits—and the remaining qubits are undisturbed. For example, suppose that a qubit is encoded via the Shor code into nine qubits and then qubit 2 and qubit 6 go missing.

Figure 235: Erasure error on 2 qubits.

It turns out that the Shor code can handle any two erasure errors (this is under the assumption that it's known on which qubits the erasure errors occurred; qubits 2 and 6 in the above example). From any seven of the nine encoded qubits, the data can be recovered.

Finally, here's a theorem that's very easy to prove but it helps clarify the distinction between redundancy and copying.

Theorem 35.1 (non-existence of a 4-qubit code protecting against two erasure errors)**.** *There does not exist a 4-qubit code that protects against two erasure errors.*

Proof. Suppose that we had such a code. Then, take the first two qubits and think of them as an encoding with two missing qubits. Same with the last two qubits.

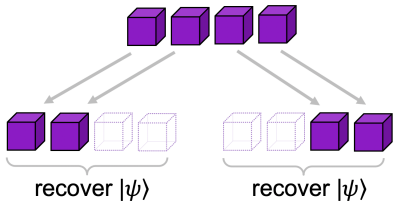

Figure 236: 4-qubit code protecting against two erasure errors violates the no-cloning theorem.

If the code was resilient against two erasure errors then we could recover the data from each of these, thereby producing two copies of the data. This would contradict the no-cloning theorem. ■

36 Calderbank-Shor-Steane codes

In this section, we will begin with a quick overview of classical linear error-correcting codes, and then I'll explain how to construct good *quantum* error-correcting codes from certain classical linear codes using a method due to Steane,[55] and Calderbank and Shor.[56] These are commonly called CSS codes.

In section 35.2, I stated results about multi-bit classical error-correcting codes, without saying anything about how these codes work. We're going to begin by looking at some of the structure of classical linear codes.

36.1 Classical linear codes

Here we consider certain block codes that encode n-bit data as m-bit codewords.

Definition 36.1 (linear code)**.** An error-correcting code is *linear* if the the mapping from data to codewords is a linear mappping from $\{0,1\}^n$ to $\{0,1\}^m$, with respect to the field $\mathbb{Z}_2$. In particular, the set of codewords is a linear space.

The 3-bit repetition code from section 35.1 is a linear code, because the set of its codewords is $\{000, 111\}$, which is a 1-dimensional subspace of $\{0,1\}^3$.

Another example of a linear code is the code given by the table in figure 237.

data	codeword
000	0000000
001	0001111
010	0110011
011	0111100
100	1010101
101	1011010
110	1100110
111	1101001

Figure 237: A 7-bit classical error-correcting code.

This code linearly maps 3-bit strings of data into 7-bit codewords. The codewords are a 3-dimensional subspace of $\{0,1\}^7$. In fact, the three strings 1010101, 0110011, 0001111 (codewords for 001, 010, 100) are a basis for the 3-dimensional subspace.

[55] A. Steane (1996). "Multiple-particle interference and quantum error correction." *Proc. Royal Society London A* **452**(1954): 2551–2577. arXiv:9601029

[56] R. Calderbank and P. W. Shor (1996). "Good quantum error-correcting codes exist." *Phys. Rev. A*, **54**(2): 1098–1105. arXiv:9512032

In section 36.3, I'll show you how codes that have this linear structure (and additional properties) can be converted into quantum error-correcting codes in a systematic way. And the 7-bit code in figure 237 will serve as a running example to illustrate the construction.

Although an error-correcting code is a mapping from $\{0,1\}^n$ to $\{0,1\}^m$, its error-correcting properties can be deduced from properties of it set of codewords, and we frequently refer to a code by its set of codewords.

Definition 36.2 (Hamming distance)**.** For any two binary m-bit strings, their *Hamming distance* is defined as the number of bit positions in which they are different. That's how many bits you need to flip to convert between the two strings.

Definition 36.3 (distance of a code)**.** The *distance of a code* is the minimum Hamming distance between any two codewords. For linear codes, this is equivalent to the minimum distance from any non-zero codeword to the zero codeword.

What's the minimum distance of the 7-bit code in figure 237? Since it's a linear code, it suffices to check the minimum Hamming weight of all the non-zero codewords, which is 4. So the minimum distance is 4.

The minimum distance of a classical code is closely related to its error-correcting properties.

Theorem 36.1. *If the minimum distance of a code is d then $\lfloor \frac{d-1}{2} \rfloor$ is the number of errors that can be corrected. In other words, as long as the number of bits flipped is strictly less than $\frac{d}{2}$, they can be corrected.*

Proof. First think of the subset of the m-bit strings that are the codewords, and associate with each codeword a *neighbourhood* that consists of all m-bit strings whose distance from that codeword is strictly less than $\frac{d}{2}$. Figure 238 is a schematic illustration of the codewords (in blue) and their associated neighborhoods (in grey).

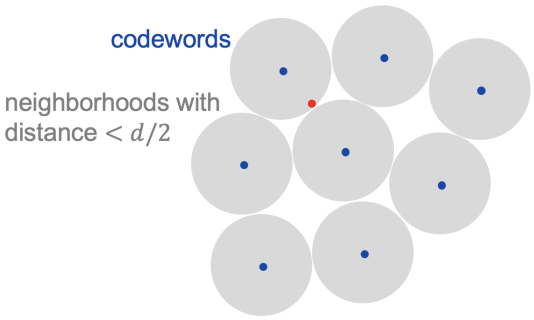

Figure 238: Neighborhoods with associated with associated with codewords.

Note that none of the neighborhoods intersect (because if they did then there would be an m-bit string whose distance from two different codewords is strictly less than $\frac{d}{2}$, violating the fact that the minimum distance is d). If fewer than $\frac{d}{2}$ bits of any codeword are flipped then the perturbed codeword stays within the same neighborhood (e.g., the red point in figure 238). Therefore, there is a unique codeword whose distance is less than $\frac{d}{2}$ from the perturbed codeword. So there is an unambiguous decoding. ■

As an example, suppose that, for the code in figure 237, one bit of a codeword is flipped, resulting in the string 1001010. Can you find the unique codeword whose distance is 1 from this string?

36.1.1 Dual of a linear code

You may recall the *dot product* between binary strings $a, b \in \{0,1\}^m$ that we previously saw in the context of Simon's algorithm, which is

$$a \cdot b = a_1 b_1 + a_2 b_2 + \cdots + a_m b_m \bmod 2. \tag{570}$$

Remember that this this is not an inner product because the dot product of a nonzero vector with itself can be zero. But it has some nice properties and we can very loosely think of two strings as being "orthogonal" if their dot product is zero.

Definition 36.4 (dual code)**.** For any linear code whose codewords are $C \subseteq \{0,1\}^m$, define the *dual code* as

$$C^\perp = \Big\{ a \in \{0,1\}^m \;\Big|\; \text{for all } b \in C,\ a \cdot b = 0 \Big\}. \tag{571}$$

The set $C^\perp$ can be loosely thought as all codewords that are orthogonal to C. The codewords of the 7-bit code in figure 237 are

$$\begin{aligned} C = \Big\{ & 0000000, 1010101, 0110011, 1100110, \\ & 0001111, 1011010, 0111100, 1101001 \Big\}, \end{aligned} \tag{572}$$

and the dual of that code is

$$\begin{aligned} C^\perp = \Big\{ & 0000000, 1010101, 0110011, 1100110, \\ & 0001111, 1011010, 0111100, 1101001, \\ & 1111111, 0101010, 1001100, 0011001, \\ & 1110000, 0100101, 1000011, 0010110 \Big\}. \end{aligned} \tag{573}$$

Every vector in C has dot product zero with every vector in $C^\perp$.

Note that $C^\perp$ is a superset of C (it contains all of C, plus additional strings). This can happen because the dot product is not an inner product. What's the minimum distance of $C^\perp$ in this example? The minimum distance of $C^\perp$ is 3. Note that this means that $C^\perp$ can correct against a 1-bit error.

36.1.2 Generator matrix and parity check matrix

For every linear code with n-bit data and m-bit codewords, we associate two matrices.

One matrix is the $n \times m$ *generator matrix*, G, which expresses the encoding process as a linear operator. Namely, for all $a \in \{0,1\}^n$,

$$E(a) = aG. \tag{574}$$

The convention in coding theory is that binary strings are row vectors and the matrix multiplication is on the right side.

For our running example of a 7-bit code, the generator matrix is

$$G = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{575}$$

and to encode a three-bit string $a \in \{0,1\}^3$, we right multiply by the generator matrix

$$E(a) = \begin{bmatrix} a_0 & a_1 & a_2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} b_0 & b_1 & b_2 & b_3 & b_4 & b_5 & b_6 \end{bmatrix}. \tag{576}$$

It's not hard to see that the rows of G are a basis for the set of codewords.

Another interesting matrix associated with a linear code is called the $m \times (m-n)$ *parity check matrix*, H, which can be used to check whether a given string is a codeword or not. For any string $b \in \{0,1\}^m$, $b \in C$ if and only if $bH = 0^{m-n}$, the zero vector. The columns of H are a basis for $C^\perp$, the dual of C.

For our 7-bit code example, a parity check matrix is

$$H = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{577}$$

Notice that the space generated by the rows of G is orthogonal to the space generated by the columns of H. This is expressed succinctly by the fact that $GH = 0^{n\times(m-n)}$, the $n \times (m-n)$ zero matrix.

36.1.3 Error-correcting via parity-check matrix

Let $C \subset \{0,1\}^m$ be a linear code with with distance d and let H be a parity check matrix of C. Here's how to correct errors using H.

For any codeword $b \in \{0,1\}^m$, let b' be the perturbed codeword after an error has been applied. It's useful to think of an m-bit *error vector*, e, that has a 1 in each position where a bit is flipped, and a 0 in the other positions. Then we can write $b' = b + e$, where the vectors are added bitwise (mod 2). If fewer than $\frac{d}{2}$ bits of b are flipped then the Hamming weight of e is less than $\frac{d}{2}$.

Now, consider what happens if we multiply b' by the parity check matrix H

$$b'H = (b+e)H \tag{578}$$
$$= bH + eH \tag{579}$$
$$= eH \tag{580}$$

(where we are using the fact that $bH = 0$ because b is a codeword).

We call eH the *syndrome of the error* e. For linear codes, the syndrome depends only on the error, and not on the codeword on which the error occurred. This property will be especially valuable when we construct quantum error-correcting codes based on classical linear codes.

Referring back to our running example 7-bit code, here's a table of the syndromes of all the error under consideration. That is, where at most one bit is flipped.

e	eH
0000000	0000
1000000	1001
0100000	1010
0010000	1011
0001000	1100
0000100	1101
0000010	1110
0000001	1111

Figure 239: Syndromes associated with errors.

In general, all errors e that are of Hamming weight less than $\frac{d}{2}$ have unique syndromes.

Let's go through am example of an error-correction procedure using the syndrome table in figure 239. Suppose that we receive the string 1001010 from the channel (and we're not told what the error is). Multiplying by the parity check matrix, we obtain $\begin{bmatrix}1\,0\,0\,1\,0\,1\,0\end{bmatrix}H = \begin{bmatrix}1\,0\,1\,1\end{bmatrix}$, so the syndrome of the error is 1011. Looking at the syndrome table, we can see that this syndrome corresponds to the error 0010000 (i.e., the third bit is flipped). So we can flip the third bit again to correct the error, obtaining the original codeword $b = 1011010$. To get the data $a \in \{0,1\}^3$ from the codeword b, we can use the fact that $\begin{bmatrix}a_0\,a_1\,a_2\end{bmatrix}G = b$ (where G is a generator matrix for the code) and solve the system of linear equations to get a.

As an aside, for large m and where d grows as a function of m, the syndrome table can be of size exponential in m. So it is not efficient to explicitly construct the entire table of errors and syndromes. Good error-correcting codes with large block sizes are designed with special additional structural properties which enable the error as a function of the syndrome to be computed efficiently.

All this information about classical linear codes is useful for understanding the methodology of CSS codes.

36.2 $H \otimes H \otimes \cdots \otimes H$ revisited

Before discussing the CSS code construction, I'd like to show you some more nice properties of the m-fold tensor product of Hadamard gates. We've already seen in section 18.3.1 of *Part II: quantum algorithms* that

$$H^{\otimes m} \left|0^m\right\rangle = \tfrac{1}{\sqrt{2^m}} \sum_{b \in \{0,1\}^m} \left|b\right\rangle, \tag{581}$$

and, more generally, for any $w \in \{0,1\}^m$,

$$H^{\otimes m} |w\rangle = \tfrac{1}{\sqrt{2^m}} \sum_{b \in \{0,1\}^m} (-1)^{b \cdot w} |b\rangle , \tag{582}$$

where $b \cdot w$ denotes the dot-product $b_0 w_0 + b_1 w_1 + \cdots + b_{m-1} w_{m-1}$.

Now, here's a generalization of the above two equations related to states that are uniform superpositions over the elements of a linear subspace $C \subseteq \{0,1\}^m$. Applying $H^{\otimes m}$ to such a state results in an equally weighted superposition of the elements of $C^\perp$. Namely,

$$H^{\otimes m} \Big(\tfrac{1}{\sqrt{|C|}} \sum_{a \in C} |a\rangle \Big) = \tfrac{1}{\sqrt{|C^\perp|}} \sum_{b \in C^\perp} |b\rangle . \tag{583}$$

Also, for a uniform superposition over the elements of a linear subspace $C \subseteq \{0,1\}^m$ offset by some $w \in \{0,1\}^m$, there is a similar expression with phases,

$$H^{\otimes m} \Big(\tfrac{1}{\sqrt{|C|}} \sum_{a \in C} |a + w\rangle \Big) = \tfrac{1}{\sqrt{|C^\perp|}} \sum_{b \in C^\perp} (-1)^{b \cdot w} |b\rangle . \tag{584}$$

Notice that Eq. (583) generalizes Eq. (581), since $\{0^m\}$ is the zero-dimensional linear subspace of $\{0,1\}^m$ and $\{0^m\}^\perp = \{0,1\}^m$. Similarly, Eq. (584) generalizes Eq. (582).

Exercise 36.1. Prove Eqns. (583) and (584).

Hint: if G is the $n \times m$ generator matrix of C then $\tfrac{1}{\sqrt{|C|}} \sum_{a \in C} |a\rangle = \tfrac{1}{\sqrt{2^n}} \sum_{b \in \{0,1\}^n} |bG\rangle$.

36.3 CSS codes

A CSS code is based on two classical linear codes, $C_0, C_1 \subseteq \{0,1\}^m$, that are related by these properties:

- $C_0 \subsetneq C_1$
- $C_0^\perp \subseteq C_1$.

Two relevant parameters are $k = \dim(C_1) - \dim(C_0)$ and d, the distance of code C_1. From this, we will construct a quantum error-correcting code that encodes k qubits as m qubits, and protects against any errors in fewer than $\frac{d}{2}$ qubits.

From our running example, we can take

$$C_0 = \Big\{0000000,\ 1010101,\ 0110011,\ 1100110,\ 0001111,\ 1011010,\ 0111100,\ 1101001\Big\}, \tag{585}$$

$$C_1 = \Big\{0000000,\ 1010101,\ 0110011,\ 1100110,\ 0001111,\ 1011010,\ 0111100,\ 1101001,\ 1111111,\ 0101010,\ 1001100,\ 0011001,\ 1110000,\ 0100101,\ 1000011,\ 0010110\Big\}. \tag{586}$$

Clearly $C_0 \subsetneq C_1$ and $C_0^\perp = C_1$. For this example, $k = \dim(C_1) - \dim(C_0) = 4 - 3 = 1$, and $d = 3$ (the distance of code C_1).

In general, let G_0 and G_1 be generator matrices for C_0 and C_1, respectively. Suppose that G_0 is an $n \times m$ matrix and G_1 is an $(n+k) \times m$ matrix. We can express

$$G_1 = \left[\frac{G_0}{W}\right], \tag{587}$$

where W is a $k \times m$ matrix representing the additional rows to add to G_0 in order to extend the span from C_0 to C_1.

In our running example, we can take

$$G_0 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \qquad W = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{588}$$

$$G_1 = \left[\begin{array}{ccccccc} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right]. \tag{589}$$

36.3.1 CSS encoding

For linear codes $C_0 \subset C_1 \subseteq \{0,1\}^m$ such that $C_0^\perp \subseteq C_1$, let $k = \dim(C_1) - \dim(C_0)$. The related CSS code encodes a k-qubit state as an m-qubit state as follows.

For all $v \in \{0,1\}^k$, define the *logical basis state* $|v\rangle_L$ as the m-qubit state

$$|v\rangle_L = \tfrac{1}{\sqrt{|C_0|}} \sum_{a \in C_0} |a + vW\rangle\,. \tag{590}$$

Then an arbitrary k-qubit (pure) state

$$\sum_{v\in\{0,1\}^k} \alpha_v |v\rangle \tag{591}$$

is encoded as the m-qubit state

$$\sum_{v\in\{0,1\}^k} \alpha_v |v\rangle_L = \sum_{v\in\{0,1\}^k} \alpha_v \Big(\tfrac{1}{\sqrt{|C_0|}} \sum_{a\in C_0} |a + vW\rangle \Big). \tag{592}$$

Returning to our running example, we have logical qubits

$$\begin{aligned} |0\rangle_L = {} & |0000000\rangle + |1010101\rangle + |0110011\rangle + |1100110\rangle \\ & |0001111\rangle + |1011010\rangle + |0111100\rangle + |1101001\rangle \end{aligned} \tag{593}$$

$$\begin{aligned} |1\rangle_L = {} & |1111111\rangle + |0101010\rangle + |1001100\rangle + |0011001\rangle \\ & |1110000\rangle + |0100101\rangle + |1000011\rangle + |0010110\rangle . \end{aligned} \tag{594}$$

The state $|0\rangle_L$ is a uniform superposition of the elements of C_0. The state $|1\rangle_L$ is a uniform superposition of the elements of C_0+1111111. Any 1-qubit state $\alpha_0 |0\rangle + \alpha_1 |1\rangle$ is encoded as the 7-qubit state $\alpha_0 |0\rangle_L + \alpha_1 |1\rangle_L$. This is called the *Steane code* and we'll show that it protects against any 1-qubit error.

36.3.2 CSS error-correcting

Let $C_0 \subset C_1 \subseteq \{0,1\}^m$ be linear such that $C_0^\perp \subseteq C_1$ and let $k = \dim(C_1) - \dim(C_0)$. We will show how to perform error-correction for the related CSS code, correcting any Pauli error acting on fewer than $\frac{d}{2}$ qubits, where d is the distance of code C_1.

We begin by showing how to correct X-errors acting on fewer than $\frac{d}{2}$ qubits. The encoding of a k-qubit state is of the form of Eq. (592). This state is a superposition of basis states from the larger code C_1, whose minimum distance is d.

We can write the X-error operator as $E_e = X^{e_0} \otimes X^{e_1} \otimes \cdots \otimes X^{e_{m-1}}$, where $e \in \{0,1\}^m$ has Hamming weight less than $\frac{d}{2}$. For encoded data $|\psi\rangle$, the state $E_e |\psi\rangle$ is the encoding subjected to the error.

Based on the parity check matrix H_1 of C_1, we can create a circuit than produces the error syndrome $|S_e\rangle$ in an ancillary register.

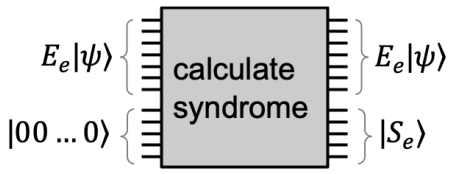

Figure 240: Computing the X-error syndrome.

Since the syndrome depends only on the error vector e and not any computational basis state from C_1, the output of the circuit is the product state $\left(E_e\,|\psi\rangle\right) \otimes |S_e\rangle$.

In our running example, this syndrome is computed using the parity check matrix

$$H_1 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{595}$$

and, based the entries of H_1, a circuit for producing the X-error syndrome is this.

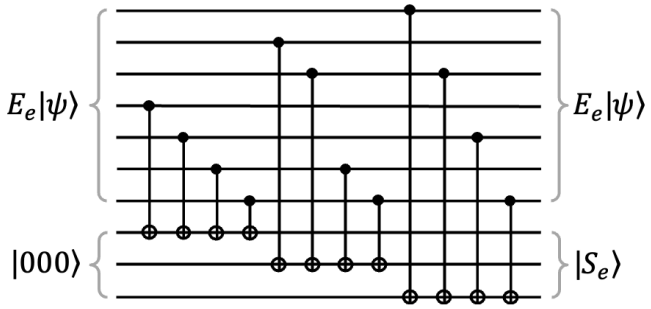

Figure 241: Explicit circuit computing the X-error syndrome for the Steane code.

It's easy to confirm that, for any computational basis state in the code word, this computes the syndrome $|S_e\rangle$ associated with error e. Once the error syndrome S_e has been computed, the error e can be deduced and undone. So this procedure corrects X-errors, as long as there are fewer than $\frac{d}{2}$ of them.

What about Z-errors? To correct against Z-errors, we apply an H operation to each qubit of the encoded data. This converts the encoding to the Hadamard basis, where Z-errors are X-errors.

What does the encoding look like in the Hadamard basis? Using the results from section 36.2, this is

$$H^{\otimes m}\left(\sum_{v\in\{0,1\}^k}\alpha_v\left|v\right\rangle_L\right)=\sum_{v\in\{0,1\}^k}\alpha_v H^{\otimes m}\left(\frac{1}{\sqrt{|C_0|}}\sum_{a\in C_0}\left|a+vW\right\rangle\right) \tag{596}$$

$$=\sum_{v\in\{0,1\}^k}\alpha_v\left(\frac{1}{\sqrt{|C_0^\perp|}}\sum_{b\in C_0^\perp}(-1)^{b\cdot(vW)}\left|b\right\rangle\right). \tag{597}$$

Since $C_0^\perp\subseteq C_1$, this state is also a superposition of computational basis states from C_1. Therefore, we can apply the procedure in figure 240 again in the Hadamard basis.

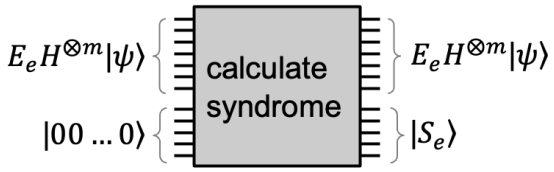

Figure 242: Computing the Z-error syndrome.

From the syndrome, the Z-errors (which are X-errors in the Hadamard basis) can be undone. Then $H^{\otimes m}$ is applied again to return to the computational basis.

So far, we can correct X-errors and Z-errors. Since $Y=iXZ$, each Y-error is like an X-error and a Z-error, and each of those is corrected by the two aforementioned procedures.

36.3.3 CSS code summary

The Steane 7-qubit CSS code and the Shor 9-qubit code both protect against a 1-qubit error; however, note that the Steane code has better rate.

In general, the performance of a CSS code depends on the performance of the classical linear codes $C_0\subset C_1\subseteq\{0,1\}^m$ (with $C_0^\perp\subseteq C_1$) on which it is based. Since good classical codes of the above form exist, there exist qualitatively good quantum error-correctng codes. It turns out that there exists a threshold $\epsilon_0=0.055...$ such that, for the depolarizing channel with error parameter $\epsilon<\epsilon_0$, there exist CSS codes of constant rate and arbitrarily small failure probability. This is qualitatively equivalent to what's achievable with classical error-correcting codes, although the specific constants are smaller.

This gives an idea of what quantum error-correcting codes can accomplish. But this is just the beginning. There are many other quantum error-correcting codes, some of which are better in certain respects than CSS codes. Also, the depolarizing channel is a kind of standard noise model, but it's not the only one. Quantum error-correcting codes that perform well against this model tend to perform well against variations of this error model. And there is some fine-tuning possible if one knows the exact error model.

37 Very brief remarks about fault-tolerance

The error-correcting codes that I've described assume that noise is restricted to the communication channel. They assume that the encoding and decoding processes are not subject to any noise. What about noise during the execution of a quantum circuit?

Suppose that we want to execute a quantum circuit, but there is noise *during the computation*. One simple way of modeling this is to assume that there is a depolarizing channel at each qubit applied at each time step.

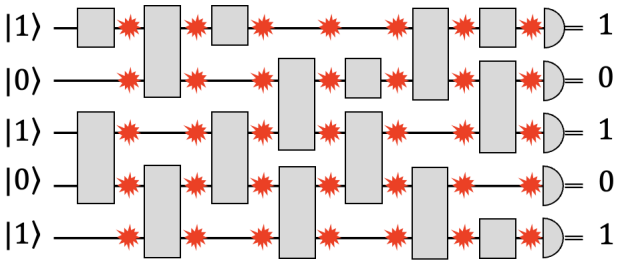

Figure 243: Noisy gates modeled by a depolorizing channel at each qubit at each time step.

How can we cope with this kind of noise? If the error parameter of the depolarizing channel ϵ is very small then this is OK. Suppose that the size of the computation is less than $\frac{1}{10\epsilon}$. Then, with good probability, none of the qubits incurs a flip (by which I mean a bit-flip, phase-flip, or both), so the computation succeeds.

But, if ϵ is a constant (dictated by the precision of our hardware), then the size of the largest circuit possible will also be bounded by a constant. If we want to execute a larger circuit then we need a smaller ϵ, which means better precision in our quantum gates. For very large circuits we would need very high precision components.

But we can do much better than this, due to the celebrated *Threshold Theorem* (building on a preliminary result by Shor,[57] which was improved by a few groups, including: Aharonov and Ben-Or,[58] and Knill, Laflamme, and Zurek[59]).

Theorem 37.1 (Threshold Theorem, rough statement)**.** *There exists a constant $\epsilon_0 > 0$ such that if the precision per gate per time step is below ϵ_0 then we can perform arbitrarily large computations without having to further improve the precision.*

[57] P. W. Shor (1996). "Fault-tolerant quantum computation." *Proc. 37th Symp. on Foundations of Computing*, pp. 56–65. arXiv:9605011

[58] D. Aharonov and M. Ben-Or (1997). "Fault-tolerant quantum computation with constant error." *Proc. 29th ACM Symp. on Theory of Computing*, pp. 176–188. arXiv:9611025

[59] E. Knill, R. Laflamme, and W. H. Zurek (1996). "Threshold accuracy threshold for quantum computation." Preprint. arXiv:9610011

The rough idea is to convert the circuit that we want to perform into a another circuit that is fault-tolerant.[60] The fault-tolerant circuit uses error-correcting codes in place of each qubit, and performs additional operations that correct errors at regular time intervals, so they don't accumulate. The known fault-tolerant constructions can be quite elaborate, and use several clever ideas. But what's nice is that the fault-tolerant circuits need not be that much larger asymptotically than the original circuits. In some formulations, the size increase is by a logarithmic factor; and in some formulations, by a constant factor.

This result is quite impressive, if you consider that there is noise during any encoding and decoding operation within the fault-tolerant circuit.

[60]The fault tolerant circuit will not be exactly of the form of a larger version of the kind of circuit that appears in Fig. 243. For the details to work out, fresh ancilla qubits must be injected into the circuit at intermediate times and also qubits must also be measured at intermediate times to perform corrections.

38 Nonlocality

In this section, I will explain a phenomenon called *nonlocality*, which is a strange kind of behavior that quantum systems can exhibit. One way of explaining this behavior is in terms of games played by a team of cooperating players. They players must individually answer certain questions, and they must do this without communicating with each other. The lack of communication appears to restrict what the players can achieve. However, with quantum information, strange behaviors can occur, that defy what one might intuitively think is possible.

38.1 Entanglement and signalling

First of all, let's note that entangled states cannot be used to communicate instantaneously. What I mean by this is the following. Suppose that Alice has a quantum system in her lab and Bob has a quantum system in his lab, and the two labs are in physically separate locations. Alice and Bob's systems might be in an entangled state—for example one of the Bell states.

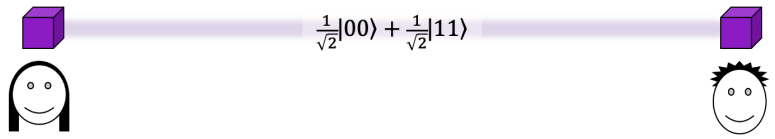

Figure 244: Is there anything Alice can do to *her* qubit that is detectable in Bob's qubit?

Then there is no quantum operation that Alice can perform on her system alone, whose effect is detectable by Bob. If Alice wants to communicate with Bob, she has to send something to him

To see why this is so, consider the density matrix of Bob's system (that is, the density matrix that results when Alice's system is traced out). If Alice performs a unitary operation or measurement on her system then this has no effect on the density matrix of Bob's system. So any kind of measurement that Bob performs on his system will have exactly the same outcome whether or not Alice performs the operation.

And this is not specific to two systems. If there are three or more parties then the same thing holds. Operations on one system have no detectable effect on the other systems.

38.2 GHZ game

Now let's consider a three-player game, commonly referred to as the GHZ game[61] (named after it's inventors, Greenberger, Horne, and Zeillinger).

Let's call the three players Alice, Bob, and Carol. This game works as follows.

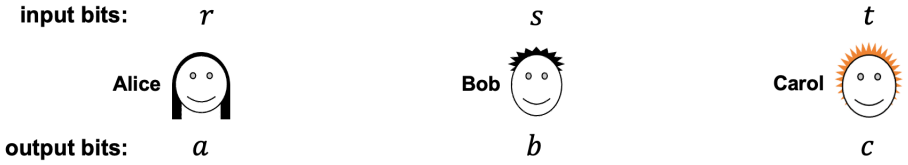

Figure 245: Alice, Bob, and Carol playing the GHZ game.

Each player receives a 1-bit input; call the respective input bits r, s, and t. And each player is required to produce a 1-bit output; call the respective input bits a, b, and c.

The rules of this game are the following.

1. It is promised that the input bits, r, s, and t have an even number of 1s among them. In other words, $r \oplus s \oplus t = 0$. So there are actually three cases of inputs: 000, 011, 101, 110.
2. There is no communication allowed between Alice, Bob, and Carol once the game starts and their input bits are received. So, for example, although Alice will know what her input r is, she does not know what s and t are.
3. This rule defines the *winning conditions* of the output bits. The three players *win* the game if and if $a \oplus b \oplus c = r \vee s \vee t$.

The condition $a \oplus b \oplus c = r \vee s \vee t$ is just a condensed way of describing this table.

rst	$a \oplus b \oplus c$
000	0
011	1
101	1
110	1

Figure 246: For each input rst, the required value of $a \oplus b \oplus c$ to win.

For the first case, the XOR of the outputs should be 0 for the players to win. For the other three cases, the XOR of the outputs should be 1 for the players to win.

[61] D. M. Greenberger, M. A. Horne, and A. Zeilinger (1989). "Going beyond Bell's Theorem," pp. 69–72, in *Bell's Theorem, Quantum Theory, and Conceptions of the Universe*, M. Kafatos (Ed.), Kluwer, Dordrecht. arXiv:0712.0921

To get a feeling for this game, here is an example of a strategy that Alice, Bob and Carol could use:

Example of a strategy

Alice receives r as input and produces r as output.

Bob receives s as input and produces $\neg s$ as output.

Carol receives t as input and produces 1 as output.

So how well does this strategy perform? Here I've added the output bits arising from this strategy to the table.

rst	$a \oplus b \oplus c$	abc
000	0	011
011	1	001
101	1	111
110	1	101

Figure 247: Output bits produced by the example strategy (in red).

You can see that the first output bit a is r, the second output bit b is $\neg s$, and the third output bit c is always 1. Consider the first case of inputs 000. There the XOR of the output bits should be 0. And it is 0. So they win in that case. For the second case, the XOR of the output bits should be 1, and it is. So they win in that case too. They also win in the third case. So far, so good. But what happens in the fourth case? In that case, the XOR should be 1. But the output bits have XOR 0. So they lose in that case.

So this is an example of a strategy that wins in three out of the four cases. Is there a better strategy, that wins in all four cases?

38.2.1 Is there a perfect strategy for GHZ?

Call a strategy that wins in every case a *perfect* strategy. Let's see if we can find a perfect strategy for this game.

Alice's output bit is a function of her input bit. Let a_0 denote her output bit if her input bit is 0. And let a_1 denote her output bit if her input bit is 1. Similarly, define b_0, b_1 as Bob's output bits in the two cases, and c_0, c_1 as Carol's output bits in the two cases. So we have six bits specifying a strategy.

We can express the winning conditions in terms of the four equations

$$a_0 \oplus b_0 \oplus c_0 = 0 \tag{598}$$
$$a_0 \oplus b_1 \oplus c_1 = 1 \tag{599}$$
$$a_1 \oplus b_0 \oplus c_1 = 1 \tag{600}$$
$$a_1 \oplus b_1 \oplus c_0 = 1. \tag{601}$$

The first equation says that, when all three inputs are 0, the XOR of the output bits should be 0. The second equation says that, when the input bits are 011, the XOR of the output bits should be 1. And so on.

So, to find a perfect strategy, we just have to solve this system of equations. Is there a solution?

In fact, there is no solution to this system of equations. These are linear equations in mod 2 arithmetic. Suppose we add the four equations. On the left side we get 0, because each variable appears exactly twice. On the right side, we get the XOR of three 1s, which is 1. So, summing the equations yields $0 = 1$, a contradiction.

It's possible to satisfy any three of the four equations, but not all four. Therefore, there does not exist a perfect deterministic strategy.

I say *deterministic*, because this analysis doesn't consider the case of probabilistic strategies. Could there exist a probabilistic perfect strategy?

There cannot exist a probabilistic perfect strategy either. This is because a probabilistic strategy is essentially a probability distribution over all the deterministic strategies, and the success probability is a weighted average of all the success probabilities of the deterministic strategies (weighted by the probabilities).

If the questions r, s, and t were selected randomly (with probability $\frac{1}{4}$ for each possible input) then the success probability for every deterministic strategy would be at most $\frac{3}{4}$. So the weighted average for any probability distribution on deterministic strategies cannot be higher than that.

Now imagine that you actually carried out this game with Alice, Bob, and Carol. You generate a random triple of questions and check whether their answers win or not. If you just play this once then they might win by luck. In fact, they can win with probability $\frac{3}{4}$. But suppose you play this several rounds in succession and they win every single round?

What if you play four rounds, once for each of the four input possibilities? Will the players necessarily fail in at least one of those rounds? No, not necessarily. Note that the players might use a different deterministic strategy at each round. Their

strategy can satisfy any three of the four equations, and they can arrange to have a different equation violated for each round.

In fact, the player can ensure that they win each round with probability $\frac{3}{4}$. So they would win any four rounds with probability $(\frac{3}{4})^4$, which is slightly more than 30%.

On the other hand, it's highly unlikely that the players would be lucky enough to win, say, 500 rounds. That success probability is $(\frac{3}{4})^{500}$, which less than 1 in a trillion trillion trillion trillion trillion.

So if you did play the game for a large number of rounds—with randomly selected questions at each round—and the players won *every single time*, what would you make of that?

38.2.2 Cheating by communicating

What makes the game non-trivial is that the players cannot communicate with each other. If they could communicate then there would be an easy perfect strategy. For example, suppose that Bob sent his input bit s to Alice. Then Alice knows r and s. She can also deduce t because of the condition that the parity of all three bits is promised to be 0. So Alice knows which of the four cases they're in. A winning strategy is for Alice to output the required parity to win, and Bob and Carol to output 0.

rst	$a \oplus b \oplus c$	abc
000	0	000
011	1	100
101	1	100
110	1	100

Figure 248: Output bits of the cheating-by-communicating strategy (in red).

This wins in all four cases. And there are ways of scrambling up the outputs that obscures this pattern—so that Bob and Carol don't always output 0, and yet the three players always win.

38.2.3 Enforcing no communication

So how can we ensure that they do not communicate? Maybe they each have a very well-concealed transmitter.

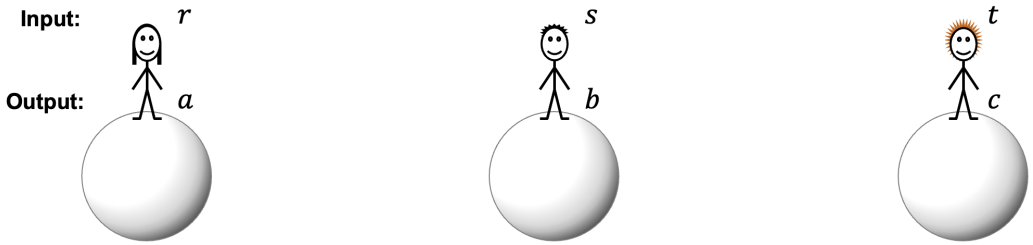

Figure 249: Alice, Bob, and Carol physically far apart.

You could ensure that there's a significant physical distance between the players (here I'm depicting them on separate planets), and also time their inputs and outputs tightly so that they cannot communicate fast enough for messages to reach each other before their deadlines for producing their outputs. For this, we assume that they cannot send signals faster than the speed of light. In physics-terminology we are making the input/output events *space-like separated*.

Then, assuming that the theory of relativity is correct, we can be sure they are not communicating their inputs to each other. But what if this is done and they *still* keep on winning, for every round?

If they players had quantum systems that were entangled, could that possibly help? Recall (as explained in section 38.1) that entanglement does not enable communication. There's no way that Bob can perform an operation on his system that can be detected by Alice. So is there any possible way that Alice, Bob, and Carol could keep on winning this game? If you're not already familiar with scenarios like this then I recommend that pause and think this over. The answer will come at the next page.

38.2.4 The "mystery" explained

The answer is yes, there is a way that they can always win, and I will show you how. They do use entanglement. Let them share the 3-qubit state

$$\tfrac{1}{2}|000\rangle - \tfrac{1}{2}|011\rangle - \tfrac{1}{2}|101\rangle - \tfrac{1}{2}|110\rangle \tag{602}$$

Alice possesses the first qubit, Bob possesses the second qubit and Carol possesses the third qubit.

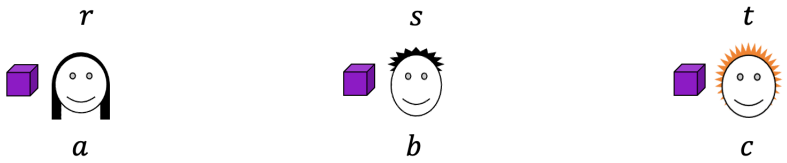

Figure 250: Alice, Bob, and Carol each possess a qubit of a tripartite entangled state.

First, I'll describe the strategy of the players.

Entangled strategy

Alice: if $r = 1$ then apply H; measure and output the result.

Bob: if $s = 1$ then apply H; measure and output the result.

Carol: if $t = 1$ then apply H; measure and output the result.

Now let's see how this strategy performs. There are four cases of inputs.

Let's begin by considering the first case, where $rst = 000$. In that case, neither player applies a Hadamard transform. Therefore, they measure the state in Eq. (602) with respect to the computational basis. The result is

$$\begin{cases} 000 & \text{with prob. } \frac{1}{4} \\ 011 & \text{with prob. } \frac{1}{4} \\ 101 & \text{with prob. } \frac{1}{4} \\ 110 & \text{with prob. } \frac{1}{4}. \end{cases} \tag{603}$$

For all four possibilities, the XOR of the three output bits is 0, which is what it's supposed to be for the 000 case.

Now let's consider the case where $rst = 011$. In that case, Bob and Carol apply Hadamard operations. It's straightforward to check that

$$\begin{aligned} \big(I \otimes H \otimes H\big)\big(\tfrac{1}{2}|000\rangle - \tfrac{1}{2}|011\rangle - \tfrac{1}{2}|101\rangle - \tfrac{1}{2}|110\rangle\big) \\ = \tfrac{1}{2}|001\rangle + \tfrac{1}{2}|010\rangle - \tfrac{1}{2}|100\rangle + \tfrac{1}{2}|111\rangle \end{aligned} \tag{604}$$

and when this state is measured in the computational basis the result is

$$\begin{cases} 001 & \text{with prob. } \frac{1}{4} \\ 010 & \text{with prob. } \frac{1}{4} \\ 100 & \text{with prob. } \frac{1}{4} \\ 111 & \text{with prob. } \frac{1}{4}. \end{cases} \tag{605}$$

So the XOR of the three output bits is 1, as required for that case.

The cases where $rst = 101$ and 110 are similar to the previous case, due to the symmetry of the state and the strategies. That's how Alice, Bob, and Carol can win with probability 1.

If they play several rounds of the game then they need to possess several copies of the entangled state in Eq. (602), and they consume one copy during each round.

38.2.5 Is the entangled strategy communicating?

So how is this entangled strategy working? Is it somehow communicating? If you look at the outcome distributions for the different cases, you can see that each individual output bit is an unbiased random bit. So the result of Alice's measurement contains absolutely no information about Bob and Carol's inputs. And similarly for the other players. In fact it can be shown that any perfect strategy using entanglement must have the property that each output bit by itself is a random unbiased bit.

Even if we consider pairs of output bits, they are uncorrelated random bits. It's only the *tripartite* correlations among all three output bits that contain information about the inputs.

38.2.6 GHZ conclusions

Let's summarize this GHZ game. It's a game played by a team of three cooperating players who cannot communicate with each other once the game starts. They each receive a bit as their input and are required to produce a bit as their output. There is a well-defined winning condition for the output bits that depends on what the input bits are.

The following conditions hold:

- Any classical team can succeed with probability at most $\frac{3}{4}$.
- Allowing the players to communicate would enable them to boost their success probability to 1.

- Entanglement cannot be used to communicate.
- Nevertheless, entanglement is another way that the players can boost their success probability to 1. But not by using entanglement to communicate.
- Instead, entanglement enables the measurement outcomes to be correlated in ways that are impossible with classical information.

You might wonder why I showed you a three-player game. Are there two-player games for which the same phenomena occurs? I showed you a three-player game because it's the simplest game that that I'm aware of that illustrates the point.

38.3 Magic square game

Here's an example of a two-player game with a property similar to that of the GHZ game: that there is no perfect classical strategy, whereas there is a perfect strategy using entanglement. It's commonly called the *Magic Square Game*, and attributed to Mermin[62] and Peres[63] and I will give just a broad overview (without explaining how the entangled strategy works).

A good way of understanding how the Magic Square Game is defined is to first consider the following puzzle. Imagine a 3-by-3 array whose entries are bits.

b_1	b_2	b_3
b_4	b_5	b_6
b_7	b_8	b_9

Figure 251: Are there bits with even parity for each row and odd parity for each column?

Can you find values for the bits such that:

(a) the number of 1s in every row is even; and

(b) the number of 1s in every column is odd?

Please pause to think about how to do this.

[62] N. D. Mermin (1990). "Simple unified form for the major no-hidden-variables theorems." *Phys. Rev. Letters*, **65**(27): 3373–3376.

[63] A. Peres (1990). "Incompatible results of quantum measurements." *Phys. Letters A*, **151**(3,4): 107–108.

You may have come to the realization that it is impossible to do this. Why? Consider the number of 1s among all the nine bits. The row condition implies that there are an odd number of 1s in total. The column condition implies that there are an even number of 1s in in total. This is a contradiction.

Now, keep this in mind, while I describe a two-player game. As with the three-player GHZ game, the players are collaborating and cannot communicate with each other once the game starts. Alice and Bob each receive a trit as input and are each required to return three bits. Think of Alice's input as specifying a row of the array, and think of Bob's input as specifying a column of the array.

Figure 252: Framework for playing the Magic Square Game.

The winning conditions are:

1. Alice's 3-bit output has even parity (think of these as the bits of one of the rows of the array).
2. Bob's 3-bit output has odd parity (think of these as the bits of one of the columns of the array).
3. Alice and Bob's outputs are consistent in the sense that, where Alice's row intersects Bob's column, the bits are the same. (For example, if Alice is queried the second row and Bob is queried the third column then Alice's third bit must be the same as Bob's second bit.)

What can we say about this game?

It turns out that there is no perfect classical strategy for this game. Of the nine possible question pairs, Alice and Bob's strategy must fail for at least one of them. The maximum success probability attainable is $\frac{8}{9}$. The proof of this is based on the fact that there is no way to set the bits of the 3-by-3 array that satisfy the parity conditions.

But there is a perfect entangled strategy that uses two Bell states as entanglement. It's a very interesting strategy, but I won't go into the details of it here.

38.4 Are nonlocal games useful?

So far, you might think that these games are weird curiosities, that have no conceivable application. But these games can be useful for enforcing certain kinds of behavior in cryptographic protocols. A simple example of this involves devices for generating random bits.

Suppose that you purchased such a device that purportedly generates a stream of random bits.

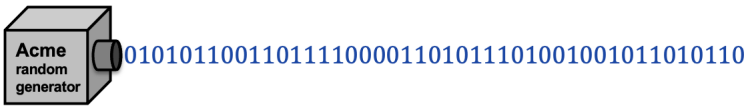

Figure 253: A supposed random number generator. Can you trust it?

How can you know that this is a good device? You could open the box and see if the internal mechanism looks legitimate. But, even if superficially it looks like some process that you think is generating randomness, it's possible that it's a fake random generator, and that the manufacturer is trying to trick you.

What the manufacturer could do is produce a long sequence of random bits in the factory and store those random bits in two memory devices, and hide one of those memory devices somewhere in the box and keep the other memory. And what the device could actually be doing is just outputting the bits stored in the memory device.

Note that the output would look like random bits to you. But the manufacturer would have a copy of those bits. They would know exactly what the next output bit is going to be. If you use such a fake random generator in a cryptographic context, for example to generate a random secret key, that that could be trouble. The manufacturer would know the key.

Unfortunately, in the context of classical information, there is no remedy for this, even in principle. If you don't trust the manufacturer of your devices, then there's no way that you can be sure that it's really generating random bits, that are unknown to the manufacturer.

But, with *quantum* devices it's possible to certify the randomness of untrusted devices. The "device" would actually consists of two components, that contain entangled quantum systems.

Figure 254: Generating random strings using untrusted devices.

You physically separate the two components and input a short random seed into each component. From this, each component outputs a long string of bits. You check if the inputs and outputs satisfy a certain function, called a test. If they pass the test, and if the input/output events are space-like separated, then you know for sure that the outputs are really random bits (within some precision ϵ, for some small $\epsilon > 0$).

I'm not claiming that such a system is practical to implement for wide usage. But it shows that, in principle, it's possible to actually certify randomness using quantum information. Something that's impossible, even in principle, with classical information.

Nonlocal games also have profound implications in the foundations of physics, which will be the topic of the next section.

39 Bell/CHSH inequality

This section is about the Bell inequality in physics, and its violation[64] (as well as subsequent variants that are all loosely called *Bell inequalities*).

39.1 Fresh randomness vs. stale randomness

I'd like to begin by making a distinction between "fresh" randomness and "stale" randomness. By *stale randomness*, I mean something like this. Suppose that I flipped a coin yesterday and I know what the outcome was, but I'm not telling you what it was. Then, from *your* perspective, the outcome is a probability distribution ("heads" with probability $\frac{1}{2}$ and "tails" with probability $\frac{1}{2}$). But the outcome is already determined. Your probabilities just reflect a lack of information on your part.

Contrast this situation with the case where the coin is spinning in the air right at this very moment. In that case, neither of us know the outcome. The outcome has not been determined yet. Let's call that *fresh randomness*.

But is a coin flip really a random process? Isn't the outcome determined by the present conditions? If we knew the exact shape of the coin, it's exact motion, and the positions of all the air molecules *and* we had an extremely powerful computer then maybe we could determine the value of the coin flip while it's spinning in the air.

Moreover, we should not conflate *forecasting* (being able to predict a future event) with *determinism* (a future event being determined by present conditions).

Consider the weather. Predicting, say, the outside temperature where I live one year for now is for all practical purposes impossible due to the chaotic nature of weather (the so-called *butterfly effect*). In terms of forecasting, this temperature is at best a probability distribution (a different distribution in the summer than in the winter). Nevertheless, it seems that the *precise* future weather is determined by the *precise* present conditions, even if we cannot know exactly what these are.

Whether the intuitive notion of fresh randomness actually exists or is just an illusion is an interesting question. I don't claim to know the answer. All of this discussion is a lead-in to the question:

> *If a qubit in state* $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ *is measured in the computational basis, will the outcome be fresh randomness or stale randomness?*

In the quantum information framework that has been the subject of these notes, the outcome is regarded as fresh randomness, that's spontaneously generated during the

[64] J. S. Bell (1964). "On the Einstein Podolsky Rosen paradox." *Physics*, **1**(3): 195–200.

measurement process. It doesn't really make sense in our model for Alice to produce a qubit in state $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and at the same time to know in advance the outcome of a future measurement of that state in the computational basis. Or does it?

39.2 Predetermined measurement outcomes of a qubit?

Let's explore the possibility that the outcomes for measuring a qubit in a state like $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ are predetermined. Think of a qubit as a physical entity, a particle (technically, a spin-$\frac{1}{2}$ particle) that was created at the big bang. Imagine that, *at the time of creation*, a predetermined outcome for each possible measurement outcome was embedded into the particle. So lurking within a qubit is some sort of table of predetermined outcomes. Let's visualize it as a literal table of outcome values.

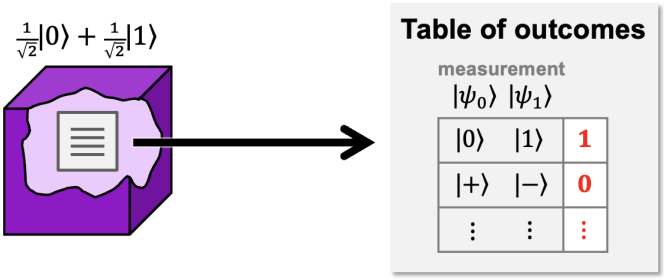

Figure 255: Are predetermined values of all measurement outcomes contained in a qubit?

Imagine that every entry of the table was created randomly so as to conform with the outcome probabilities of quantum measurements. For example, for a measurement in the computational basis, the outcome is an unbiased random bit. So, in that entry of the table, a random bit was inserted (in the figure, it was set to 1). On the other hand, for a measurement in the $|+\rangle/|-\rangle$ basis, the outcome should always be the first state $|+\rangle$. So that entry of the table was set the bit 0. And so on. For every other potential measurement, there is an entry in the table containing a bit that is sampled with the appropriate probability distribution for that measurement of the state. That's an infinitely large table. Maybe there's a compressed way of containing this information, but let's not concern ourselves with that issue. My point is that it's conceivable that the particle contains this table of predetermined measurement outcomes stored within it.

In physics, these are called *hidden variables*. The idea is that these hidden variables represent additional physical properties of systems that are yet to be discovered.

When they are discovered, quantum mechanics will be tamed of its randomness. The randomness that arises in quantum theory as it currently exists could merely be a consequence of the fact that we don't know what these hidden variables are. In this way of thinking, a measurement merely extracts a predetermined value from the table of outcomes.

Let's continue developing this model. What happens if we apply a unitary operation to a qubit? This would rearrange the table of outcomes in some systematic way. For example, suppose that we apply a Hadamard transform to the qubit. That would swap the first two bits of the table. This is because, after applying a Hadamard to this $|+\rangle$ state, the state becomes $|0\rangle$, so now a measurement in the computational basis produces 0 for sure. And measuring in the $|+\rangle/|-\rangle$ basis is what produces a random bit. Without going into the details, the effect of any unitary operation can be captured by moving around the entries of the table of outcomes. Unitary operations merely rearrange the stale randomness of the table of outcomes.

And, in this picture, every spin-$\frac{1}{2}$ particle has it's own separate table. If multiple particles are in the $|+\rangle$ state then each one contains an independent random bit for the first entry in its table. This is consistent with what happens when we measure several qubits that are each in the $|+\rangle$ state.

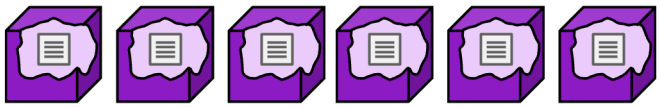

Figure 256: Multiple qubits, each containing a "table" of hidden variables.

What's interesting about this local hidden variables picture is that, so far, everything is consistent with quantum behavior.

We might imagine that this model can be extended to capture all of quantum information theory. For example, when a 2-qubit unitary gate entangles two particles, what might actually be happening is a rearrangement of both tables of outcomes, so that the entries are appropriately correlated for all possible measurements. If the unitary operation creates the state $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ then the table entries for measuring each qubit in the computational basis should be: 0 for both particles with probability $\frac{1}{2}$; and 1 for both particles with probability $\frac{1}{2}$.

Let's continue exploring how a local hidden variable model would work.

39.3 CHSH inequality

A few years after Bell's ground-breaking paper,[64] Clauser, Horne, Shimoney, and Holt discovered a particularly elegant variant of it that is now commonly referred to as the *CHSH inequality*.[65] I will explain this result here.

Imagine a system consisting of two qubits (or two particles), and that there are two measurements, that we'll refer to as M_0 and M_1. We're supposing that each particle contains a full tables of outcomes, but here we'll only care about the parts of the tables that are associated with the measurements M_0 and M_1.

Figure 257: Two particles, with their hidden variables for two measurements, M_0 and M_1.

Call the two predetermined values for the first particle a_0 and a_1. And call the two predetermined values for the second particle b_0 and b_1.

Note that we making an assumption that the hidden variables are *local* hidden variables in the sense that each particle's predetermined outcomes depend only on the measurement performed on that particle, and not the measurements performed on the other particles. What's the justification for this?

The justification is that the particles might be far apart from each other and the timing of the measurements might be such that the two measurement events are space-like separated. This means that there isn't sufficient time for a signal to go from the second particle to the first particle with the information about which measurement is performed. For space-like separated measurement events, the first particle has no way of "knowing" what measurement is being performed on the second particle, even in principle. Therefore, for space-like separated measurements, it's impossible for one particle's outcome to depend on what measurement is performed on the other particle.

[65] J. F. Clauser, M. A. Horne, A. Shimony, and R. A. Holt (1969). "Proposed experiment to test local hidden-variable theories." *Phys. Rev. Letters*, **23**(15): 880–884.

I will now describe a property that the bits a_0, a_1, b_0, b_1 must satisfy. It is convenient to describe this property in terms of bits that are expressed as $+1$ and -1 instead of 0 and 1. So we define such a conversion, into "uppercase" bits as

$$A_0 = (-1)^{a_0} \tag{606}$$
$$A_1 = (-1)^{a_1} \tag{607}$$
$$B_0 = (-1)^{b_0} \tag{608}$$
$$B_1 = (-1)^{b_1}. \tag{609}$$

I claim that inequality (610) holds, which is called the CHSH inequality.

Theorem 39.1 (CHSH inequality)**.** *For any $A_0, A_1, B_0, B_1 \in \{+1, -1\}$, it holds that*

$$A_0B_0 + A_0B_1 + A_1B_0 - A_1B_1 \leq 2. \tag{610}$$

Note that each of the four terms on the left side can be $+1$ or -1. So we might imagine that the left side can be as large as 4. But the upper bound is 2.

Proof of Theorem 39.1. Let's see how to prove the CHSH inequality. We can write the left side of Eq. (610) as

$$A_0(B_0 + B_1) + A_1(B_0 - B_1). \tag{611}$$

Now consider the expressions in the parentheses, $B_0 + B_1$ and $B_0 - B_1$. Either B_0 and B_1 have the same sign or they have different signs. If B_0 and B_1 have the same sign then $B_0 + B_1$ can be as large as 2, but then $B_0 - B_1 = 0$. So in that case, the upper bound is 2. If B_0 and B_1 have the different signs then $B_0 - B_1$ can be as large as 2, but then $B_0 + B_1 = 0$. So in that case, the upper bound is also 2. This completes the proof. ■

Why should we care about this CHSH inequality? The reason why is that the inequality can be experimentally tested, and if an experiment shows that it's violated then the possibility of a local hidden variable model is refuted.

First of all, let's consider how one could in principle design an experiment to verify that systems satisfy the CHSH inequality. There's some subtlety with this. The problem is that, for any single measurement of the two particles, only one of the four A_sB_t-terms in the inequality can be measured.

Here again are the particles with their outcome tables, where I've taken the liberty of writing the outcomes with uppercase bits (in the ± 1 language).

Figure 258: Two particles, with their hidden variables for M_0 and M_1 specified as $\pm$ bits.

You can choose to perform any single measurement on the first system and get either A_0 or A_1 and then the state is disturbed, so you cannot measure again to get the original value of the other bit. Similarly, you can choose any single measurement on the second system and get B_0 or B_1 (but not both). So you can acquire only one of the four terms A_0B_0, A_0B_1, A_1B_0, A_1B_1 (whose value is $+1$ or -1). To verify the inequality, you would need to see all four terms.

However, the Bell inequality *can* be verified using statistical methods, by making several independent runs, using a separate pair of particles for each run. In each run, $st \in \{00, 01, 10, 11\}$ is chosen randomly and the $\pm$ bit $(-1)^{s \cdot t} A_s B_t$ is calculated. The factor $(-1)^{s \cdot t}$ means multiply by -1 in the $st = 11$ case.

If $st \in \{00, 01, 10, 11\}$ is sampled randomly according to the uniform distribution then the *expected value* of $(-1)^{s \cdot t} A_s B_t$ is

$$\mathrm{E}_{s,t}\Big[(-1)^{s \cdot t} A_s B_t\Big] = \tfrac{1}{4} A_0 B_0 + \tfrac{1}{4} A_0 B_1 + \tfrac{1}{4} A_1 B_0 - \tfrac{1}{4} A_1 B_1. \tag{612}$$

Does this expression look familiar? It's the left side of the CHSH inequality divided by 4. So we can deduce from the CHSH inequality that

$$\mathrm{E}_{s,t}\Big[(-1)^{s \cdot t} A_s B_t\Big] \leq \tfrac{1}{2}. \tag{613}$$

The experiment to statistically verify the CHSH inequality is to make many separate runs, each on a separate pair of particles, of the procedure where you pick a random $st \in \{00, 01, 10, 11\}$ and then measure M_s and M_t to create a sample $(-1)^{s \cdot t} A_s B_t \in \{+1, -1\}$. If local variables exist then the average over many runs should converge to $\frac{1}{2}$ or less.

Note that, in order to eliminate the possibility of a hidden variable model that is *not local*, the experiment should be implemented so that each pair of measurement

events is space-like separated. If they are not space-like separated then the experiment does not eliminate the possibility that nature is behaving in a conspiratorial way, with signaling between pairs of particles, that permits the outcomes for each particle to depend on both measurements.

The fact that the CHSH inequality can be experimentally verified is remarkable because ...

39.4 Violating the CHSH inequality

... quantum systems can violate the CHSH inequality!

To see how, suppose that the two physically separated qubits are entangled in the Bell state $\frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle$.

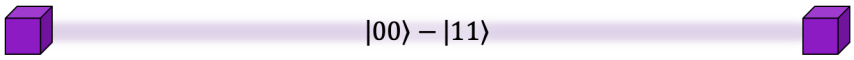

Figure 259: Two physically separated particles in the Bell state $\frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle$.

Consider what happens if a rotation is performed on each qubit, by angle θ_s for the first qubit and θ_t for the second qubit.

Exercise 39.1 (straightforward)**.** Check that, if $R(\theta_s) \otimes R(\theta_t)$ is applied to state $\frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle$ then the result is

$$\cos(\theta_s+\theta_t)\Big(\tfrac{1}{\sqrt{2}}|00\rangle - \tfrac{1}{\sqrt{2}}|11\rangle\Big) + \sin(\theta_s+\theta_t)\Big(\tfrac{1}{\sqrt{2}}|01\rangle + \tfrac{1}{\sqrt{2}}|10\rangle\Big). \tag{614}$$

If the state in Eq. (614) is measured in the computational basis then the outcome bits ($a_s, b_t \in \{0,1\}$) satisfy

$$\Pr[a_s \oplus b_t = 0] = \cos^2(\theta_s+\theta_t) \tag{615}$$

$$\Pr[a_s \oplus b_t = 1] = \sin^2(\theta_s+\theta_t). \tag{616}$$

It follows that, for the $\pm$ bits $A_s = (-1)^{a_s}$ and $B_t = (-1)^{b_t}$,

$$\begin{aligned}\mathrm{E}[A_sB_t] &= \cos^2(\theta_s+\theta_t) - \sin^2(\theta_s+\theta_t)\\ &= \frac{1+\cos\Big(2(\theta_s+\theta_t)\Big)}{2} - \frac{1-\cos\Big(2(\theta_s+\theta_t)\Big)}{2}\\ &= \cos\Big(2(\theta_s+\theta_t)\Big).\end{aligned} \tag{617}$$

Now define the measurements M_0 and M_1 as follows.

- M_0: rotate by $\theta_0 = -\frac{\pi}{16}$ and then measure in the computational basis.
- M_1: rotate by $\theta_1 = +\frac{3\pi}{16}$ and then measure in the computational basis.

Let's look at the various angles $\theta_s + \theta_t$ that arise for $st \in \{00, 01, 10, 11\}$.

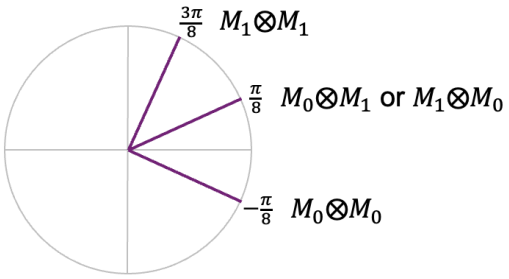

Figure 260: Angles $\theta_s + \theta_t$ that arise for $M_s \otimes M_t$, for the cases $st \in \{00, 01, 10, 11\}$.

When M_0 is performed on both sides, $\theta_0 + \theta_0 = -\frac{\pi}{8}$. When M_0 is performed on one side and M_1 on the other side, $\theta_0 + \theta_1 = \theta_1 + \theta_0 = +\frac{\pi}{8}$. And when M_1 is performed on both sides, $\theta_1 + \theta_1 = \frac{3\pi}{8}$.

Applying Eq. (617), this means that, for measurements M_s and M_t, the $\pm$ outcomes A_s and B_t have the property that

$$\mathrm{E}[A_s B_t] = \begin{cases} \cos(\pm\frac{\pi}{4}) & \text{if } st \in \{00, 01, 10\} \\ \cos(\frac{3\pi}{4}) & \text{if } st = 11 \end{cases} \tag{618}$$

$$= \begin{cases} \frac{1}{\sqrt{2}} & \text{if } st \in \{00, 01, 10\} \\ -\frac{1}{\sqrt{2}} & \text{if } st = 11. \end{cases} \tag{619}$$

It follows that

$$\mathrm{E}[(-1)^{st} A_s B_t] = \tfrac{1}{2}\sqrt{2}, \tag{620}$$

which clearly violates the CHSH inequality—explicitly the upper bound in Eq. (613).

So if we performed the aforementioned experiment of repeatedly picking a random $st \in \{00, 01, 10, 11\}$ and applying the measurements M_s and M_t to a pair of qubits in state $\frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle$ to sample $(-1)^{s \cdot t} A_s B_t$ then the average would exceed the bound of $\frac{1}{2}$, that was derived under the assumption of local hidden variables. Therefore, in the quantum information framework, local hidden variables cannot exist.

Summary and experimental implementations

Let's summarize the Bell inequality and its violation. Assuming that the measurement outcomes of quantum systems are predetermined by local hidden variables leads to the Bell inequality. But actual quantum quantum systems violate this inequality, by a factor of $\sqrt{2}$. Therefore, quantum systems cannot be based on local hidden variables.

And this behavior of quantum systems has been experimentally verified. The rough idea is to generate two particles in a Bell state and send them out in opposite directions to reach detectors, which are set to measure the particles in various ways.

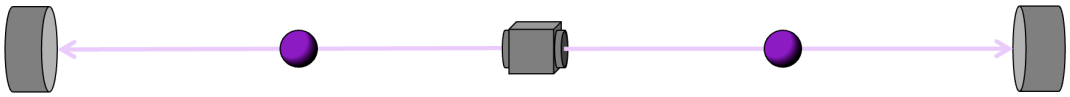

Figure 261: Form of test of the CHSH inequality violation.

In order for such an experiment to be *loophole-free*, the measurement events must be space-like separated; the precision in the state preparation and the measurements must exceed certain thresholds, and the random choices of $st \in \{00, 01, 10, 11\}$ must actually be random. This is non-trivial, but such experiments that are widely regarded as loophole-free have been performed, refuting the existence of local hidden variables.

39.5 Bell/CHSH inequality as a nonlocal game

Now let's look at the Bell/CHSH inequality and its violation in a different way, as a nonlocal game, similar to the ones that we saw in sections 38.2 (the GHZ game) and 38.3 (the Magic Square game).

Define the *CHSH game* as follows. Alice and Bob receive input bits, s and t, and they must produce output bits, a and b.

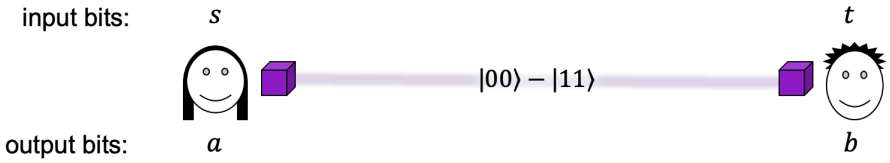

Figure 262: Alice and Bob playing the CHSH game.

The rules of the game are as follows. First, a question pair $st \in \{00, 01, 10, 11\}$ is randomly selected according to the uniform distribution. As usual for nonlocal games,

there is no communication allowed between the players once the game starts. And the players *win* if and only if $a \oplus b = s \wedge t$, which is just a condensed way of specifying the following table.

st	$a \oplus b$
00	0
01	0
10	0
11	1

Figure 263: For each input st, the required value of $a \oplus b$ to win.

What's interesting about the CHSH game is that:

- The maximum winning probability for any *classical* strategy is $\frac{3}{4}$.
- There exists an *entangled* strategy that wins with probability

$$\cos^2\left(\tfrac{\pi}{8}\right) = \tfrac{1}{2}\left(1 + \tfrac{1}{\sqrt{2}}\right) = 0.853... \tag{621}$$

This is essentially the CHSH inequality and its violation, but where the outputs are $\{0, 1\}$-bits instead of $\{+1, -1\}$-bits. The upper bound on the winning probability corresponds to the CHSH inequality (in Eq. (613)), and the quantum strategy that attains success probability $\cos^2\left(\frac{\pi}{8}\right)$ corresponds to the CHSH inequality violation (in section 39.4). However, I will provide a separate analysis of this game.

We can analyze classical strategies for the CHSH game in a manner similar to the way we analyzed the GHZ game in section 38.2.1. First note that any deterministic strategy can be described by four bits, a_0, a_1 (Alice's output bits for the two input possibilities), b_0, b_1 (Bob's output bits for the two input possibilities). And the winning condition can be expressed as the four equations

$$a_0 \oplus b_0 = 0 \tag{622}$$
$$a_0 \oplus b_1 = 0 \tag{623}$$
$$a_1 \oplus b_0 = 0 \tag{624}$$
$$a_1 \oplus b_1 = 1. \tag{625}$$

It's easy to show that at most three of these four equations can be satisfied, so the maximum success probability of any deterministic strategy is $\frac{3}{4}$. Since any classical

probabilistic strategy is essentially a probability distribution on the set of all deterministic strategies, its winning probability cannot be higher than $\frac{3}{4}$.

The entangled strategy for the CHSH game whose probability of winning is $\cos^2(\frac{\pi}{8})$ is very similar to the CHSH inequality violation in section 39.4. Alice and Bob use the entangled state $\frac{1}{\sqrt{2}}|00\rangle - \frac{1}{\sqrt{2}}|11\rangle$ and they each perform the following on their qubit.

```
1: if input bit is 0 then
2:     apply R(−π/16) to qubit
3: else if input bit is 1 then
4:     apply R(+3π/16) to qubit
5: end if
6: measure qubit and output result
```

Figure 264: Alice and Bob's local behavior based on their input bit.

From Eqns. (614)(615)(616), we can deduce that, for input bits s and t, Alice and Bob's output bits a and b satisfy

$$\Pr[a \oplus b = s \wedge t] = \begin{cases} \cos^2\left(\pm\frac{\pi}{8}\right) & \text{if } st \in \{00, 01, 10\} \\ \sin^2\left(\frac{3\pi}{8}\right) & \text{if } st = 11 \end{cases} \tag{626}$$

$$= \cos^2\left(\frac{\pi}{8}\right). \tag{627}$$

Bell inequalities and nonlocal games can be thought of as different ways of expressing the same ideas about nonlocality. One perspective is to consider (and refute) the existence of local hidden variables. Another perspective is to consider communication protocols between separated parties, and what they can accomplish with and without the resource of entanglement.