

Introduction to Quantum Information Science Lecture Notes

Scott Aaronson¹

Fall 2018

¹With crucial help from: Corey Ostrove and Paulo Alves

Contents

	Page
1 Course Introduction and The Extended Church-Turing Thesis	7
2 Probability Theory and Quantum Mechanics	11
2.1 Linear Algebra Approach to Probability Theory	15
3 Basic Rules of Quantum Mechanics	19
3.1 Quantum States and The Ket Notation	19
3.2 Transforming Quantum States	21
3.3 Quantum Interference	22
3.3.1 Global and Relative Phase	23
4 Quantum Gates and Circuits, Quantum Zeno and The Elitzur-Vaidman Bomb	25
4.1 Quantum Gates	25
4.1.1 Generalized Born Rule	26
4.1.2 General Properties of Quantum Gates and Measurements	26
4.2 Quantum Circuit Notation	29
4.3 Quantum Zeno Effect	30
4.4 The Elitzur-Vaidman Bomb	32
5 The Coin Problem, Distinguishability, Multi-Qubit States and Entanglement	34
5.1 The Coin Problem	34
5.2 Distinguishability of Quantum States	35
5.3 Multi-Qubit States and Operations	37
5.3.1 Multi-Qubit Operations	37
5.3.2 Entanglement	39

6	Mixed States	42
6.1	Mixed States	42
6.1.1	Density Matrices	43
6.1.2	Properties of Density Matrices	46
6.1.3	Partial Trace and Reduced Density Matrices	47
7	The Bloch Sphere, No-Cloning Theorem and Wiesner’s Quantum Money Scheme	50
7.1	The Bloch Sphere	50
7.1.1	Quantum Gates in the Bloch Sphere Representation	53
7.2	The No-Cloning Theorem	54
7.3	Quantum Money	56
7.3.1	Wiesner’s Quantum Money Scheme	56
8	Quantum Money and Quantum Key Distribution	58
8.1	Quantum Money Attacks	58
8.1.1	Interactive Attacks	59
8.1.2	Public-Key Quantum Money	61
8.2	Quantum Key Distribution	61
9	Superdense Coding	65
9.1	Superdense Coding	65
10	Teleportation, Entanglement Swapping, GHZ State and The Monogamy of Entanglement	67
10.1	Quantum Teleportation	67
10.1.1	Multi-Qubit Teleportation and Entanglement Swapping	70
10.1.2	The GHZ State and Monogamy of Entanglement	72
11	Quantifying Entanglement	75
11.1	Schmidt Decomposition	76
11.2	Von Neumann Entropy	76
11.2.1	Entanglement Entropy	77
11.3	Mixed State Entanglement	79
12	Interpretations of Quantum Mechanics	82
12.1	The Copenhagen Interpretation	83
12.1.1	“Shut Up and Calculate”	83
12.2	Schrödinger’s Cat and Wigner’s Friend	84
12.3	Dynamical Collapse	85
12.3.1	Ghirardi-Rimini-Weber (GRW) Theory	87

12.3.2 Penrose Theory	87
12.4 The Many-Worlds Interpretation	90
13 Hidden Variables and Bell's Inequality	96
13.1 Hidden Variable Theories	96
13.1.1 Bohmian Mechanics	96
13.1.2 Local Hidden Variable Theories	98
13.2 The CHSH Game	99
14 Nonlocal Games	103
14.1 CHSH Game: Quantum Strategy	103
14.1.1 Analysis of Protocol	104
14.1.2 CHSH Game: Interpretations and Local Realism	106
14.1.3 Tsirelson's Inequality	107
14.1.4 Experimental Tests of Bell's Inequalities	109
14.2 The Odd Cycle Game	111
14.3 The Magic Square Game	113
15 Einstein-Certified Randomness	115
15.1 Guaranteed Random Numbers	115
15.1.1 Leashing Quantum Systems	119
16 Quantum Computing and Universal Gate Sets	120
16.1 Complexity of General Unitaries: Counting Argument	123
16.2 Universal Gate Sets	125
16.2.1 Classical Universality	125
16.2.2 Quantum Universality	127
16.2.3 The Solovay-Kitaev Theorem	129
17 Quantum Query Complexity and The Deutsch-Josza Problem	131
17.1 Quantum Query Complexity	132
17.2 Quantum Garbage Collection	134
17.3 Deutsch's Algorithm	137
17.4 Deutsch-Josza Algorithm	138
18 Bernstein-Vazirani and Simon's Algorithm	141
18.1 The Bernstein-Vazirani Problem	141
18.1.1 Quantum Algorithm	141
18.2 Simon's Problem	143
18.2.1 Classical Lower Bound	145
18.2.2 Quantum Algorithm	146

19 RSA and Shor's Algorithm	150
19.1 RSA Encryption	150
19.2 Period Finding	152
19.2.1 Factoring to Period-Finding Reduction	153
19.2.2 Quantum Algorithm for Period-Finding	156
20 Quantum Fourier Transform	158
20.1 Quantum Fourier Transform	158
20.1.1 Implementing the QFT	161
20.1.2 Period Finding Using the QFT	164
21 Continued Fractions and Shor's Algorithm Wrap-Up	168
21.1 Continued Fraction Algorithm	169
21.2 Applications of Shor's Algorithm	171
21.2.1 Graph Isomorphism	172
21.2.2 Lattice-Based Cryptography	172
22 Grover's Algorithm	174
22.1 The Algorithm	176
22.1.1 Implementing the Diffusion Operator	179
22.1.2 Geometric Interpretation	180
22.1.3 Analysis	181
22.1.4 Multiple Marked Items	184
23 BBBV Theorem and Applications of Grover's Algorithm	188
23.1 The BBBV Theorem	188
23.2 Applications of Grover's Algorithm	194
23.2.1 OR of ANDs	194
24 More Grover Applications and Quantum Complexity Theory	198
24.1 More Applications of Grover's Algorithm	198
24.1.1 The Collision Problem	198
24.1.2 Element Distinctness	200
24.2 Parity Lower Bound	202
24.3 Quantum Complexity Theory	203
25 Hamiltonians	206
25.1 Quantum Algorithms for NP-complete Problems	206
25.2 Hamiltonians	207
25.2.1 Matrix Exponentiation	208
25.2.2 Energy	211

25.2.3	Tensor Products of Hamiltonians	215
25.2.4	Addition of Hamiltonians	215
26	The Adiabatic Algorithm	220
26.1	Local Hamiltonians	220
26.2	The Adiabatic Algorithm	222
27	Quantum Error Correction	233
27.1	Classical Error Correction	234
27.1.1	Classical Fault-Tolerance	236
27.2	Quantum Error Correction	237
27.2.1	The Shor 9-Qubit Code	241
27.2.2	Quantum Fault Tolerance	246
28	The Stabilizer Formalism	248
28.1	The Gottesman-Knill Theorem	252
28.1.1	The Gottesman-Knill Algorithm	253
28.2	Stabilizer Codes	256
28.2.1	Transversal Gates	258

Lecture 1: Course Introduction and The Extended Church-Turing Thesis

- ▶ Quantum Information Science is an inherently interdisciplinary field (Physics, CS, Math, Engineering, Philosophy)
- ▶ It's not just about inventing useful devices and algorithms, but also about clarifying the workings of quantum mechanics.
 - We use it to ask questions about what you can and can't do with quantum mechanics
 - It can help us better understand the nature of quantum mechanics itself.
- ▶ Professor Aaronson is very much on the theoretical end of research.
- ▶ Theorists inform what experimentalists make, which in turn informs theorists' queries

Today we'll articulate several “self-evident” statements about the physical world. We'll then see that quantum mechanics leaves some of these statements in place, but overturns others—with the distinctions between the statements it upholds and the ones it overturns often extremely subtle! To start with...

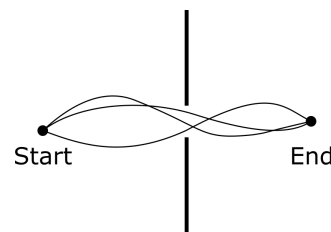
Probability ($P \in [0, 1]$) is the standard way of representing uncertainty in the world. Probabilities have to follow certain axioms such as:

- ▶ Given a set of n mutually exclusive exhaustive events, the sum of the probabilities satisfies $P_1 + P_2 + \dots + P_n = 1$
- ▶ The probability of any particular event satisfies $P_i \geq 0$

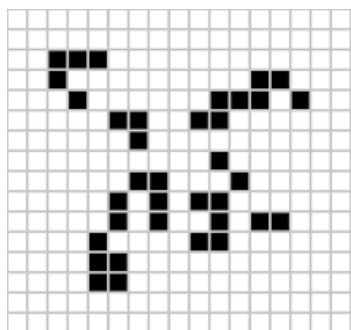
•
•
•

There's a view that "probabilities are all in our heads." Which is to say that if we knew everything about the universe (let's say position/velocity of all atoms in the solar system) that we could just crunch the equations and see that things either happen or they don't.

Let's suppose we have two points separated by a barrier with an open slit, and we want to measure the probability that a particle goes from one point to the other. It seems obviously true that increasing the number of paths (say, by opening another slit) should increase, or at any rate not decrease, the likelihood that it will reach the other end. We refer to this property by saying that probabilities are **monotone**.



Locality is the idea that things can only propagate through the universe at a certain speed. When we update the state of a little patch of space, it should only require knowledge of a small neighborhood around it. Conway's Game Of Life (left) is a good model here: changes you make to the system can affect it, but since each cell only directly interacts with its nearest neighbors the changes only propagate at a certain speed.



In physics, locality naturally emerges due to Einstein's Special Theory of Relativity which implies that no signal can propagate faster than the (finite) speed of light; this simple principle can explain a large number of physical phenomena. In special relativity anything traveling faster than the speed of light would be effectively traveling backwards in time, from some observer's standpoint.

Local Realism is the principle that any instantaneous update in knowledge about faraway events can be explained by correlations of random variables. For example, if you in Austin and a friend in San Francisco both subscribe to the same newspaper then when you read your copy in the morning your knowledge of the headline on your friend-in-San-Francisco's copy instantly collapses to whatever your copy's headline is. Before picking up the copy in the morning your knowledge of the headline for yourself and your friend may have been best described by a probability

distribution over various possibilities, but since the outcomes are perfectly correlated, as soon as you learn the headline on your copy you instantly know that your friend's must be the same.

Some popular science articles talk about how if you measure the spin of one particle then instantaneously you can know the spin of another particle on the other side of the galaxy. But unless and until something more is said about it, that's no different from the case of the newspapers and seems 100% compatible with local realism!

Church-Turing Thesis The Church-Turing Thesis states that every physical process can be simulated by a Turing machine to any desired precision. The way that Church and Turing understood this was as a definition of computation, but we can think of it instead as a falsifiable claim about the physical world. You can think about this as the idea that the entire universe is sort of a gigantic video game: you've got all sorts of complicated things, like quarks and black holes and whatnot, but at the end of the day you've got to be able to simulate it on a computer. The **Extended Church-Turing Thesis** says moreover that, when we simulate reality on a digital computer, there's at most a polynomial (e.g., linear or quadratic) blowup in time, space, and other computational resources.

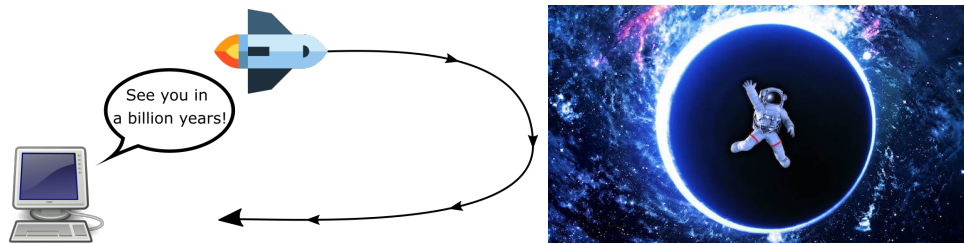
Theoretical computer science courses can be seen as basically math courses. So what does connect them to reality? The Church-Turing Thesis.

So, what does quantum mechanics have to say about each of these principles? To give you a teaser for much of the rest of the course:

- ▶ We'll still use probabilities. But the way we'll *calculate* probabilities will be totally different, and will violate the axiom of monotonicity. That is, *increasing* the number of ways for an event to happen, can *decrease* the probability that it happens.
- ▶ Locality will be upheld. But *Local Realism* will be overthrown. And if those two principles sounded like restatements of each other—well, quantum mechanics will dramatically illustrate the difference between them!

- ▶ As we'll see, the Church-Turing Thesis still seems to be in good shape, even in light of quantum mechanics, but the Extended Church-Turing Thesis seems to be false, with quantum computing standing as a glaring counterexample to it—possibly the one counterexample that our laws of physics allow. With that said, however, one can formulate a quantum version of the Extended Church-Turing Thesis, which remains true as far as anyone knows today.

*You could imagine other possible counter-examples to the Extended Church-Turing Thesis. For instance, some have proposed that by using time dilation you could travel billions of years in the future and get results to hard problems. Fun! But you'd need a **LOT** of energy, and if you have that much energy in one place you collapse to a black hole. Not so fun!*



Lecture 2: Probability Theory and Quantum Mechanics

The famous theoretical physicist Richard Feynman said that everything about quantum mechanics could be encapsulated in the **Double Slit Experiment**. In the double-slit experiment, you shoot photons one at a time toward a wall with two narrow slits. Where each photon lands on a second wall is probabilistic. If we plot where photons appear on the back wall, some places are very likely, some not. In Figures 2.1 – 2.3 you can see diagrams showing the basic experimental set-up and results from performing both single-slit and double-slit experiments with photons.

Note that some places on the screen being likely and others unlikely in and of itself isn't the weird part: we could totally explain this by some theory where each photon just had some extra degree of freedom (an "RFID tag") that we didn't know about, and that determined which way it went. What's weird is as follows. For some interval on the second wall:

Let P be the probability that the photon lands in the interval with both slits open.

Let P_1 be the probability that the photon lands in the interval if only slit 1 is open.

Let P_2 be the probability that the photon lands in the interval if only slit 2 is open.

You'd think that $P = P_1 + P_2$. But experiment finds that that's not the case! Even places that are never hit when both slits are open, can sometimes be hit if only one slit is open.

The weirdness isn't that "God plays dice," but rather that "these aren't normal dice"!

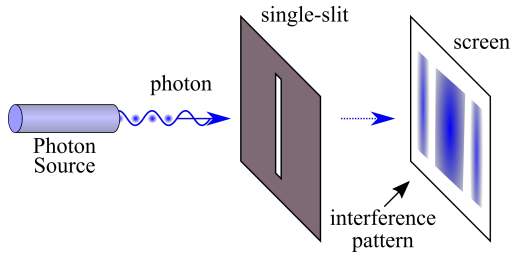


Figure 2.1: Experimental setup for a *single-slit* photon interference experiment

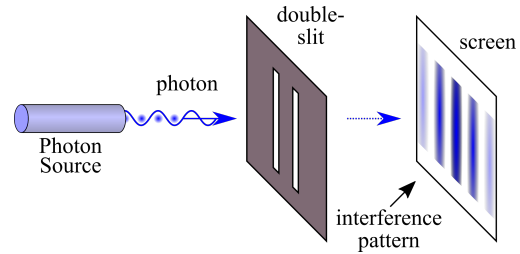


Figure 2.2: Experimental setup for a *double-slit* photon interference experiment

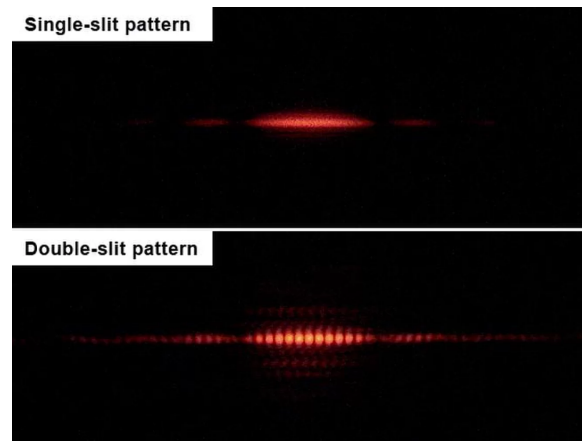


Figure 2.3: Comparison of single-slit and double-slit interference patterns as seen in an actual experiment. Notice in the double-slit pattern the appearance of new dark spots (areas with a low probability of a photon landing there) not seen for the single-slit.

You may think to measure which slit the photon went through, but doing so *changes* the measurement results into something that makes more sense, with just two bright patches, one for each slit. Note that it isn't important whether there's a conscious observer: if the information about which slit the photon went through leaks out in any way into the outside environment, the results go back to looking like they obey classical probability theory.

As if Nature says "What? Me? I didn't do anything!"

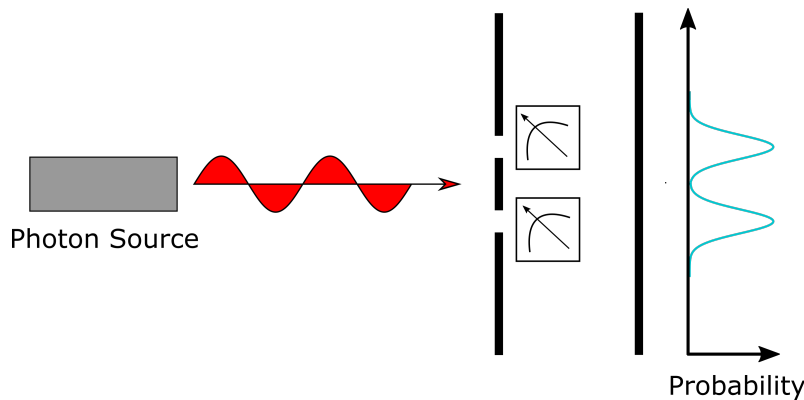


Figure 2.4: Double-slit experiment with measuring devices on each slit which measure which slit any given photon passes through. In this case the probability distribution looks like the average of the individual single slit distributions.

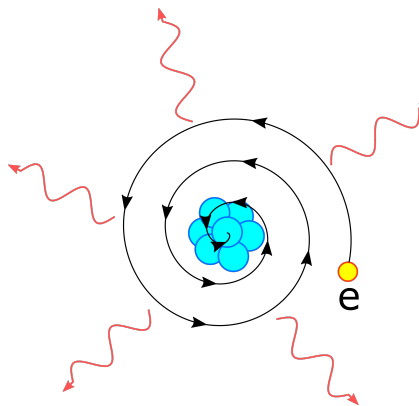
This reversion to classical probability theory when systems are coupled to their environments is called **decoherence**. Decoherence is why the usual laws of probability look like they work in everyday life. A cat isn't found in a superposition of alive and dead states, because it interacts constantly with its environment. These interactions essentially leak information about the 'cat system' out. Quantum superposition is something that happens to particles, or groups of particles, when they're isolated from their environments. Needing the particles to be isolated is why it's so hard to build a quantum computer.

*And what if the particles aren't **perfectly** isolated, but merely **mostly** isolated? Great question! We'll come back to it later in the course.*

The story of atomic physics between roughly 1900 and 1926 is that scientists kept finding things that didn't fit with the usual laws of mechanics or probability. They often came up with hacky solutions that explained a

phenomenon without connecting it to much else. That is, until Heisenberg, Schrödinger, etc. came up with the general rules of quantum mechanics.

Briefly, though, take the usual high school model of the electron, rotating around a nucleus in a fixed orbit. Scientists realized that this model would mean that the electron, as an accelerating electric charge, would be constantly losing energy in the form of radiation and spiraling inwards until it hit the nucleus. To explain how the electron orbits could remain stable, along with the double-slit experiment and countless other phenomena, physicists eventually had to change the way probabilities were calculated.



Instead of using probabilities $P \in [0, 1]$, they started using **amplitudes** $\alpha \in \mathbb{C}$. Amplitudes can be positive or negative, or more generally complex numbers (with real and imaginary parts). *The central claim of quantum mechanics* is that to fully describe the state of an isolated system, you need to give one amplitude for each possible configuration that you could find the system in on measuring it.

The **Born Rule** says that the probability you see a particular outcome is the squared absolute value of the amplitude:

$$P = |\alpha|^2 = \text{Re}(\alpha)^2 + \text{Im}(\alpha)^2 \quad (2.1)$$

So let's see how amplitudes being complex leads them to act differently from probabilities. Let's revisit the double-slit experiment using amplitudes. We'll say that:

Let α the total amplitude of a photon landing in a certain spot on the screen,

α_1 the amplitude it lands in the spot if only slit 1 is open and,

α_2 the amplitude it lands in the spot if only slit 2 is open.

By analogy to classical probability we have

$$\alpha = \alpha_1 + \alpha_2 \quad (2.2)$$

Then from the Born rule, Equation 2.1, we have

$$P = |\alpha|^2 = |\alpha_1 + \alpha_2|^2 = |\alpha_1|^2 + |\alpha_2|^2 + \alpha_1^* \alpha_2 + \alpha_1 \alpha_2^*. \quad (2.3)$$

If, for example, $\alpha_1 = 1/2$ and $\alpha_2 = -1/2$, then we find $P = 0$ if both slits are open but, $P = 1/4$ if only one slit is open; This phenomenon is known as **interference**.

So then, to justify the electron not spiraling into the nucleus we can say that, yes, there are many paths where the electron does do that, but some have positive amplitudes and others have negative amplitudes and they end up canceling each other out.

With some physics we won't cover in this class, you'd discover that the possibilities where amplitudes don't cancel each other out lead to discrete energy levels, which are the places where the electrons can sit. The phenomenon of discrete energy levels is in turn what leads to chemistry.

2.1 Linear Algebra Approach to Probability Theory

We use **Linear Algebra** to model the states of systems as vectors and the evolution of systems in isolation as transformations of vectors. In the simplest case, for a system with two states, we could write

$$M \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} a'_1 \\ a'_2 \end{bmatrix}. \quad (2.4)$$

For now, we'll consider classical probability. Let's look at flipping a coin. We model this with a vector assigning a probability to each possibility: $p = P(\text{heads})$ and $q = P(\text{tails})$.

$$\begin{bmatrix} p \\ q \end{bmatrix} \quad \begin{array}{l} p, q \geq 0 \\ p + q = 1 \end{array} \quad (2.5)$$

We can apply a transformation, like turning the coin over.

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} q \\ p \end{bmatrix} \quad (2.6)$$

Turning the coin over means the probability that the coin *was* heads is now the probability that the coin *is* tails. If it helps, you can think of the

transformation matrix as:

$$\begin{bmatrix} P(\text{heads}|\text{heads}) & P(\text{heads}|\text{tails}) \\ P(\text{tails}|\text{heads}) & P(\text{tails}|\text{tails}) \end{bmatrix} \quad (2.7)$$

where $P(a|b)$ is the conditional probability for the state of the coin to be “a” given that it was previously in the state “b.” We could also flip the coin fairly.

$$\begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} = \begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} \quad (2.8)$$

Which means that regardless of previous state, both possibilities are now equally likely. Let’s say we flip the coin, and if we get heads we flip again, but if we get tails we turn it to heads.

$$\begin{bmatrix} 0 & \frac{1}{2} \\ 1 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} \frac{q}{2} \\ p + \frac{q}{2} \end{bmatrix} \quad (2.9)$$

Does this make sense? Since if the state of the coin is found to be heads we do a fair flip we can see that given it’s heads the probability after we do the flip for heads or tails is $\frac{1}{2}$. That is, $P(\text{heads}|\text{heads}) = P(\text{tails}|\text{heads}) = \frac{1}{2}$. If we see a tails however we always flip it back to heads and so $P(\text{tails}|\text{tails}) = 0$ and $P(\text{tails}|\text{heads}) = 1$.

So, which matrices can be used as transformations? Firstly, we know that all entries have to be non-negative (because probabilities can’t be negative). We also know that each column must sum to 1, since we need the sum of initial probabilities to equal the sum of the transformed probabilities (namely, both should equal 1). A matrix that satisfies these conditions is called a **Stochastic Matrix**.

Now let’s say we want to flip two coins, or rather, two bits. For the first bit $a = P(0)$ and $b = P(1)$. For the second let $c = P(0)$ and $d = P(1)$.

$$\begin{matrix} 0 \\ 1 \end{matrix} \begin{bmatrix} a \\ b \end{bmatrix} \quad \begin{matrix} 0 \\ 1 \end{matrix} \begin{bmatrix} c \\ d \end{bmatrix}$$

To combine the two vectors we need a new operation, called **Tensor Product**:

$$\begin{bmatrix} a \\ b \end{bmatrix} \otimes \begin{bmatrix} c \\ d \end{bmatrix} = \begin{bmatrix} P(00) \\ P(01) \\ P(10) \\ P(11) \end{bmatrix} = \begin{bmatrix} ac \\ ad \\ bc \\ bd \end{bmatrix} \quad (2.10)$$

It's worth noting that not all possible 4-element vectors can arise by the tensor product of two 2-element vectors. For example, suppose by contradiction that

$$\begin{bmatrix} ac \\ ad \\ bc \\ bd \end{bmatrix} = \begin{bmatrix} \frac{1}{2} \\ 0 \\ 0 \\ \frac{1}{2} \end{bmatrix}. \quad (2.11)$$

Then multiplying the first and last equations together implies that $(ac)(bd) = \frac{1}{4}$, while multiplying the second and third implies that $(ad)(bc) = 0$, giving a contradiction. As such, the 4-element vector on the right hand side can't be written as the tensor product of two 2-element vectors.

As we did with the one-bit systems, we can describe probabilistic transformations of two-bit systems using stochastic matrices. For example, let's say we apply a transformation where if the first bit is 1 then we flip the second bit and otherwise we do nothing to the system. The 4×4 matrix that achieves this is given in Equation 2.12, and is called the Controlled NOT or CNOT matrix; it will also come up often in quantum computing.

$$\text{CNOT} = \begin{array}{c} \begin{matrix} & 00 & 01 & 10 & 11 \end{matrix} \\ \begin{matrix} 00 \\ 01 \\ 10 \\ 11 \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array} \quad (2.12)$$

Suppose we apply the CNOT matrix to the following vector, representing a system where the first bit is either zero or one with $\frac{1}{2}$ probability and the second bit is always 0:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{2} \\ 0 \\ \frac{1}{2} \\ 0 \end{bmatrix} = \begin{array}{c} 00 \\ 01 \\ 10 \\ 11 \end{array} \begin{bmatrix} \frac{1}{2} \\ 0 \\ 0 \\ \frac{1}{2} \end{bmatrix} \quad (2.13)$$

We see that we've reached an output distribution that we previously proved can't arise as a tensor product! Such a distribution is called **correlated**: learning one bit tells you something about the other bit. In this case, the two bits are always equal; with 50% probability they're both 0 and with 50% probability they're both 1. So, we've learned that the CNOT matrix can *create*

correlations: it can transform an uncorrelated distribution into a correlated one.

Quantum mechanics essentially follows the same process to model states using vectors, *except* that it uses amplitudes instead of probabilities. We can describe transformations of vectors representing quantum states using matrices.

$$\begin{bmatrix} U \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \quad (2.14)$$

where instead of preserving the sum of the vector entries we preserve the sum of the squared-amplitudes $\sum_{i=1}^3 |a_i|^2 = \sum_{i=1}^3 |b_i|^2 = 1$, which by Equation 2.1 (the Born rule) implies that the probabilities of the outcomes sum to 1. Such matrices are called **Unitary Matrices**, and in the next lecture we'll describe their properties as well as the basic mathematical rules for handling quantum states in much greater detail.

Lecture 3: Basic Rules of Quantum Mechanics

3.1 Quantum States and The Ket Notation

A *quantum state* (technically, a “pure state”) is a unit vector in \mathbb{C}^N describing the state of a quantum system. The dimension, N , could theoretically be anything. Physics courses cover infinite-dimensional quantum states such as position or momentum states, but we’ll stick to discrete systems (which is to say that when we make a measurement, there are only finitely many possible outcomes).

What does quantum mechanics say about the universe being discrete or continuous at its most basic level? It suggests a strange, hybrid picture. Even for a simple two dimensional system there’s a continuum of possible quantum states, but every measurement has a discrete outcome. A system with two amplitudes, $[\alpha, \beta]^T$, has uncountably infinitely many possible states (even with the restriction that $|\alpha|^2 + |\beta|^2 = 1$), though note that the same would be true even if we described states using classical probabilities. In both cases, classical and quantum, as long as we stick to finite-dimensional systems the continuum is never directly observed, but is only used in calculating the probabilities of discrete outcomes.

A *qubit* is the simplest interesting quantum system. It’s a two-level system (we label the levels “0” and “1”), with an amplitude for 0 and an amplitude for 1.

A one-level quantum system would just be [1]. Not very

interesting!

Following the notation introduced by Paul Dirac, we write quantum state vectors using the so-called **Ket Notation**

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \alpha |0\rangle + \beta |1\rangle = |\psi\rangle.$$

Note that $|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ and $|1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, and that we'll often use $|\psi\rangle$ to refer generically to a quantum state.

Why do we use the ket notation? One main advantage is that practically speaking, we often deal with really sparse vectors (where most amplitudes are 0). Ket notation makes it easy to represent only the values we're talking about.

It's really just a formalism to make life easier, we can put anything in ket notation. Look, this is Schrödinger's Cat in ket notation: $\frac{1}{\sqrt{2}} (|\text{😄}\rangle + |\text{😞}\rangle)$.

Another motivation for adopting the ket notation is that it simplifies many of the linear algebra operations we use frequently in quantum mechanics. Often, for example, we'll need to take the transpose (or conjugate transpose for complex-valued vectors):

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} \rightarrow [\alpha^* \quad \beta^*]$$

and we can use these conjugate-transposed vectors to define a norm on our vector space

$$\|v\|^2 = [\alpha^* \quad \beta^*] \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = |\alpha|^2 + |\beta|^2 \quad (3.1)$$

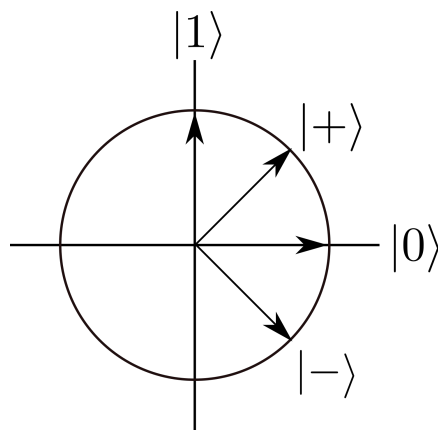
In ket notation both of these operations can easily be represented. The conjugate-transpose of a ket, $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$, is represented by a corresponding object called a **bra** $\langle\psi|$ where

$$\langle\psi| = \alpha^* \langle 0| + \beta^* \langle 1|.$$

We can also use the bra as we've just defined it to write the inner product between a pair of kets, $|x\rangle$ and $|y\rangle$, as $\langle x|y\rangle$. The norm of the ket $|\psi\rangle$ in this notation is $\| \psi \|^2 = \langle \psi | \psi \rangle$. The inner product as it's been defined here has the

property that $\langle x|y\rangle = \langle y|x\rangle^*$. Note that we'll be adopting the physics convention of denoting the conjugate-transpose with the \dagger symbol (read dagger).

The set of all possible pure quantum states of a qubit with real coefficients defines a circle. The set of all possible quantum states with complex coefficients defines a sphere known as the Bloch sphere, which we will learn about in greater detail in a later lecture. In addition to the states $|0\rangle$ and $|1\rangle$ (known as the “standard basis states”) there are four other single qubit states that occur so frequently in quantum information theory that they've been given special names:



$$\begin{aligned}
 |+\rangle &= \frac{|0\rangle + |1\rangle}{\sqrt{2}} \\
 |-\rangle &= \frac{|0\rangle - |1\rangle}{\sqrt{2}} \\
 |i\rangle &= \frac{|0\rangle + i|1\rangle}{\sqrt{2}} \\
 |-i\rangle &= \frac{|0\rangle - i|1\rangle}{\sqrt{2}}
 \end{aligned} \tag{3.2}$$

Figure 3.1: Standard and Hadamard basis states represented on the real plane.

The pair $\{|+\rangle, |-\rangle\}$ is often referred to as the “Hadamard basis.” Figure 3.1 shows the positions of the standard and Hadamard basis states in the real plane.

3.2 Transforming Quantum States

Just like with classical probability theory, one basic way to change quantum states is by applying linear transformations.

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

In order for a linear transformation to correspond to a valid transformation of a quantum state, we require that it be **unitary**. A transformation on a single

qubit, U , is unitary if $|\alpha|^2 + |\beta|^2 = |\alpha'|^2 + |\beta'|^2$ for all input vectors $[\alpha, \beta]^\top$. In other words, U preserves the 2-norm of the vector.

Examples of 1-Qubit Unitary Transformations

$$\begin{array}{cccc}
 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} & \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \\
 \text{Identity} & \text{NOT Gate} & \text{Relative Phase Shift} & \text{2D-Rotations}
 \end{array} \quad (3.3)$$

The second-to-last unitary matrix above has the effect of mapping $|0\rangle \rightarrow |0\rangle$ and $|1\rangle \rightarrow i|1\rangle$; in fact, we can replace the i in this matrix with any unit magnitude complex number of the form $e^{i\theta}$. Remember Euler's equation,

$$e^{i\theta} = \cos(\theta) + i \sin(\theta). \quad (3.4)$$

Rotations also preserve the 2-norm of vectors and so for example, we can use the last matrix above to rotate our state in the real plane by some specified angle θ (we denote this family of matrices R_θ).

Since a unitary matrix, U , preserves the 2-norm of vectors it immediately follows that it must also preserve the value of the inner product $\langle\psi|\psi\rangle$. This gives the following series of equalities $\langle\psi|\psi\rangle = (|\psi\rangle)^\dagger |\psi\rangle = (U|\psi\rangle)^\dagger U|\psi\rangle = \langle\psi|U^\dagger U|\psi\rangle$. This can only be true for all $|\psi\rangle$ if $U^\dagger U = I$, which implies that for a unitary matrix $U^{-1} = U^\dagger$. It also implies that the rows of U must be an orthogonal unit basis. Conversely, it's easy to see that if $U^{-1} = U^\dagger$ then U is unitary. So you can tell if a matrix is unitary by checking if $U^\dagger U = I$, or equivalently if the rows (or the columns) form an orthogonal unit basis.

This is not the “operational definition” of unitary matrices, but is a logical consequence of unitary transformations preserving the 2-norm.

An **orthogonal matrix** is both unitary and real-valued. Any orthogonal matrix is a product of rotations and reflections. For example, the matrix $R_{\pi/4}$ is orthogonal. Applying $R_{\pi/4}$ repeatedly to the input state $|0\rangle$ gives the sequence of states $|0\rangle \rightarrow |+\rangle \rightarrow |1\rangle \rightarrow -|-\rangle \dots$. You'll get a full revolution after applying $R_{\pi/4}$ eight times.

3.3 Quantum Interference

In the classical world, if an event could happen multiple ways, but will be “random” no matter which way it happens, then it's simply “random” overall.

But in the quantum world, you can sometimes apply a unitary transformation to a superposition state and get a determinate answer, even though the answer would have been random had you applied it to any individual component of the superposition. Many of the most interesting phenomena in quantum mechanics can be explained in terms of *quantum interference*. As an illustrative example, suppose we start initially in the $|0\rangle$ state. We then apply twice a unitary transformation which when applied to $|0\rangle$ places us in $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and which when applied to $|1\rangle$ places us in $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$. This situation drawn in Figure 3.2.

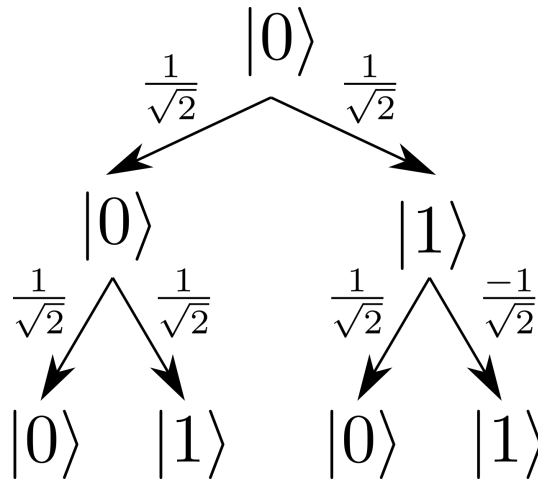


Figure 3.2: Diagram showing effect of applying a unitary operation which maps $|0\rangle \rightarrow |+\rangle$ and $|1\rangle \rightarrow |-\rangle$.

To get the amplitude associated with a particular path we take the product of the amplitudes along that path. Then to get the final amplitude associated with a particular *output*, we sum the amplitudes for each of the paths through the tree leading to that output. In this case, the amplitude of $|0\rangle$ is 1 and the amplitude of $|1\rangle$ is 0. The paths leading to $|0\rangle$ interfere constructively while the paths leading to $|1\rangle$ interfere destructively.

3.3.1 Global and Relative Phase

No matter what unitary transformation you apply, because it is a linear operation we always have that $U(-|0\rangle) = -U|0\rangle$ or more generally $U(c|0\rangle) = cU|0\rangle$ for any constant c . When c is a unit magnitude complex number (i.e. it can be written in the form $e^{i\theta}$ for some θ) we call it a *global phase*. The states $|\psi\rangle$ and $e^{i\theta}|\psi\rangle$ are physically indistinguishable, which is to say: Global phase

is unobservable! Multiplying your entire quantum state by a scalar is like if last night someone moved the entire universe twenty feet to the left. We can only really measure things relative to other things!

But this leads to a second maxim: Relative phase is observable. To distinguish between the states $|+\rangle$ and $|-\rangle$ which were defined in Equation 3.1, for example, we can rotate by 45 degrees (by applying $R_{\pi/4}$, perhaps) and then measure to see whether we got $|0\rangle$ or $|1\rangle$ (see Figure 3.1). The relative phase difference between the states $|+\rangle$ and $|-\rangle$ is observable precisely because there is a sequence of unitary operations and measurements (an experiment) that we can perform to distinguish between the two cases. For global phases there is no experiment one could ever do which would distinguish between $|\psi\rangle$ and $e^{i\theta}|\psi\rangle$.

Lecture 4: Quantum Gates and Circuits, Quantum Zeno and The Elitzur-Vaidman Bomb

4.1 Quantum Gates

In quantum information theory we often refer to small unitary transformations as “gates.” We use some of these gates so often that they have special names and symbols associated with them. For example, the matrix $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ is called the NOT gate. One can also define a $\sqrt{\text{NOT}}$ gate,

$$\sqrt{\text{NOT}} = \frac{1}{2} \begin{bmatrix} 1+i & 1-i \\ 1-i & 1+i \end{bmatrix}, \quad (4.1)$$

which you can check satisfies the property that $\sqrt{\text{NOT}}\sqrt{\text{NOT}} = \text{NOT}$. One of the most ubiquitous gates in quantum information is the **Hadamard gate**

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (4.2)$$

The Hadamard gate is so useful because it maps the $\{|0\rangle, |1\rangle\}$ basis to the $\{|+\rangle, |-\rangle\}$ basis, and vice versa.

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

Similarly, $H|1\rangle = |-\rangle$, $H|+\rangle = |0\rangle$ and $H|-\rangle = |1\rangle$. Note that the $\{|0\rangle, |1\rangle\}$ basis and $\{|+\rangle, |-\rangle\}$ basis form different two different orthogonal (and complementary) bases with the special property that being maximally

certain in the $\{|0\rangle, |1\rangle\}$ basis means that you're maximally uncertain in the $\{|+\rangle, |-\rangle\}$ basis and vice versa. Another example of a basis changing gate is

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ i & -i \end{bmatrix}. \quad (4.3)$$

This one switches us from the $\{|0\rangle, |1\rangle\}$ basis to the $\{|i\rangle, |-i\rangle\}$ basis. *Why would we want to use multiple bases?* We like to think of vectors existing abstractly in a vector space, but to do computations we often need to pick a convenient basis. When we see some actual quantum algorithms and protocols, we'll see the power that comes from switching between bases.

4.1.1 Generalized Born Rule

We can define quantum measurement more generally. Measuring the state $|\psi\rangle$ in the orthonormal basis $\{|V_0\rangle, \dots, |V_{N-1}\rangle\}$, you'll get the outcome $|V_i\rangle$ with probability $|\langle V_i|\psi\rangle|^2$. So the probability of the outcome $|V_i\rangle$ is the squared length of the projection onto that basis vector.

When talking about the Born Rule in earlier lectures, we've been using the special case of the $\{|0\rangle, |1\rangle\}$ basis for simplicity.

To implement a measurement in an arbitrary basis, you can use unitary transformations to convert between bases. So for example, to measure in the $\{|V_0\rangle, \dots, |V_{N-1}\rangle\}$ basis, you can first apply a unitary transformation U such that $U|V_0\rangle = |0\rangle$, $U|V_1\rangle = |1\rangle$, etc. . . , and then measure in the standard basis $\{|0\rangle, \dots, |N-1\rangle\}$.

There's an extreme point of view in quantum mechanics that unitary transformations are the only thing that really exist, and measurements don't. And the converse also exists: the view that measurements are the only things that really exist, and unitary transformations don't. More about this when we talk about interpretations!

4.1.2 General Properties of Quantum Gates and Measurements

Unitary Transformations are:

- ▶ Invertible: This should be clear, since preserving the 2-norm means that $U^\dagger U = I$ which means $U^{-1} = U^\dagger$.
 - In other words, the transformation $|\psi\rangle \rightarrow U|\psi\rangle$ can always be reversed by applying U^\dagger , since $U^\dagger U|\psi\rangle = |\psi\rangle$.

Interestingly this implies that unitary evolution can never destroy information, which should imply that the universe is reversible. Physics has treated the microscopic laws as reversible since Galileo's time (e.g. a time-reversed video of a swinging pendulum still shows it obeying the laws of physics). So for example burning a book shouldn't destroy the information within, as physics says that in principle you can recover all the information from the smoke and ash left over.

- ▶ Deterministic: There is nothing probabilistic in the unitary evolution process itself.
- ▶ Continuous: Unitary transformations take place over intervals of time which can always be broken into smaller and smaller subintervals.

This last item is part of why it's important that unitary matrices are in general complex-valued. If, for example, the transformation $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ was applied by some process which took 1 second, then by applying the same process for half of a second, we can obtain $\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$ or some other square root of the transformation. But we invite you to check that $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ has no 2×2 real-valued square root.

By the way, if we allow ourselves the ability to add an extra dimension then there *is* a 3×3 matrix that “squares” to $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}^2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}.$$

But, to take a square root of $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$, either you need complex numbers, or else you need to add a third dimension. The latter is analogous to reflecting

your three-dimensional self by rotating yourself in a fourth dimension—as in some science fiction stories!

Important: If you come back reflected after a trip into the fourth dimension, don't eat anything without first consulting medical professionals. Normal food will have molecules of the wrong chirality for you to digest them.

Measurements break all three rules of unitary transformations! Measurements are:

- ▶ Irreversible: Whatever information about the system you didn't capture is now lost.
- ▶ Probabilistic: Everything in quantum mechanics is deterministic *until* measurement, but measurement outcomes are in general random.
- ▶ Discontinuous: The “collapse of the amplitude vector” is conventionally treated as an instantaneous event.

So how can we reconcile these two sets of rules? That's the famous **Measurement Problem**; we'll talk about various points of view on it later. Despite the philosophical conflict, unitary transformations and measurement sync up well because unitary transformations preserve the 2-norm and measurement gives probabilities determined by the 2-norm.

Classical probability is based on the 1-norm, while quantum mechanics is based on the 2-norm. So it's natural to wonder: what about theories based on the 3-norm, 4-norm, etc...? Actually, there don't seem to be any interesting theories there, making quantum mechanics a bit of “an island in theory space.” If you try to adjust anything about it in any way, you typically get junk! You could alternatively say that there seems to be “nothing near quantum mechanics, that's nearly as nice as quantum mechanics itself.” Another example of this are some of the (somewhat technical) reasons why complex numbers work better than the reals, or for that matter quaternions, as amplitudes.

4.2 Quantum Circuit Notation

Quantum Circuit Notation gives us a graphical language to keep track of which qubits we have, which operations we're applying to them and in which order. In a quantum circuit diagram we use wires to represent qubits, and labeled boxes and other symbols to represent unitary transformations on those qubits. When reading a quantum circuit diagram we interpret time (typically thought of as occurring in discrete time-steps) as flowing from left to right. Say for example we want to represent a single qubit, initialized to the $|1\rangle$ state, which has two consecutive Hadamard gates applied to it, followed by a measurement in the standard basis. This can be represented using the circuit diagram below.

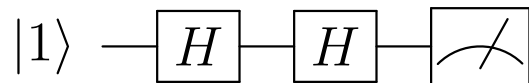


Figure 4.1: Quantum circuit representing two Hadamard gates followed by a measurement on a qubit initialized to the $|1\rangle$ state.

The operations in Figure 4.1 are simple enough that we'd have no trouble writing out the matrices and vectors explicitly, but as we add more qubits to our system and allow for multi-qubit operations, matrix representations quickly become unwieldy (see the final two gates in Table 4.1). Quantum circuits give us a tool for succinctly describing all manner of complicated quantum transformations.

Quantum circuits allow for operations on an arbitrary number of qubits. Here is a circuit containing a two-qubit gate, labeled U , which is followed by a Hadamard on the first qubit and then measurements on both qubits.

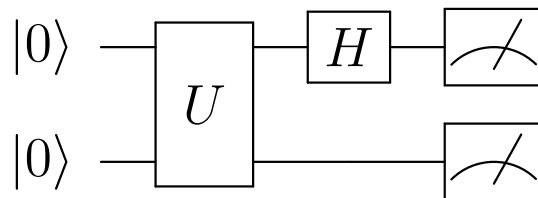


Figure 4.2: A generic two qubit operation followed by a Hadamard on the first qubit and a pair of measurements.

In some contexts it is useful to allow our circuits to have some additional “workspace,” perhaps to save intermediate results in a calculation. To enlarge

a system in this manner we can add new qubits (typically all assumed to be initialized to $|0\rangle$) to the system which are called ancilla qubits.

One final notational convention we'll introduce in this section is for controlled gates. This includes the CNOT gate that we saw first in Section 2.1. A controlled gate can be split into two parts, the control and the target. We represent a control qubit using a thick solid dot on a wire. We then draw a vertical line connecting to a gate on another qubit(s) which we wish to control. In the figure below a pair of arbitrary qubits (meaning we won't specify the input ahead of time) has a series of controlled gates applied.

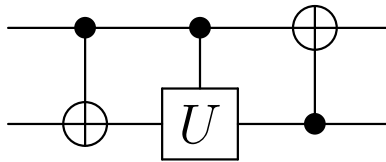
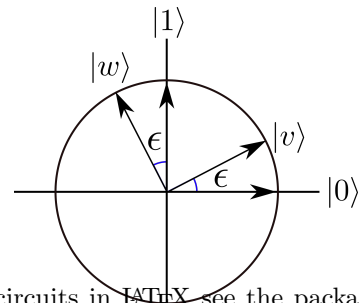


Figure 4.3: Different types of controlled operations. The first gate is a CNOT with the first qubit as the control and the second qubit as the target. Notice the special notation (\oplus) for the target of the CNOT gate. The second is a controlled-U operation, where U is arbitrary. This operation applies U if the control qubit is $|1\rangle$ and does nothing otherwise. The final gate is a CNOT gate with second qubit as the control and the first as the target. Control can run in either direction up or down.

There are other notational conventions used for various gates relevant to quantum information, more than we can go through in detail. For a summary of the most common ones that we'll come across during the course see Table 4.1.¹

4.3 Quantum Zeno Effect

There are several interesting phenomena that already happen in the quantum mechanics of one qubit. Suppose we have a qubit in the state $|0\rangle$. and let's say we want to put it in the $|1\rangle$ state without using any unitary transformations. For some small ϵ , we can measure the qubit in a basis



¹Also, for more information on typesetting quantum circuits in L^AT_EX see the package “qcircuit” on CTAN. <https://ctan.org/tex-archive/graphics/qcircuit>


















Gate	Common Name	Circuit Representation	Matrix
NOT/ X	Pauli-X or NOT	 or 	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Z	Pauli-Z or Phase-Flip		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Y	Pauli-Y		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
H	Hadamard		$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
S	Phase Gate		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
T	T Gate		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
$\sqrt{\text{NOT}}$	Square-Root of NOT		$\begin{bmatrix} 1+i & 1-i \\ 1-i & 1+i \end{bmatrix}$
R_θ	Rotation Gate		$\begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$
R_ϕ	Phase-Shift		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}$
CNOT	Controlled-NOT or CNOT		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
CPHASE/CZ	Controlled-Z	 or  or 	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP	Swap Gate		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
CCNOT	Toffoli or Controlled-Controlled-NOT		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$
C-SWAP	Fredkin or Controlled-SWAP		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

Table 4.1: Circuit representations for some commonly used gates in quantum information theory.

that's rotated from the $\{|0\rangle, |1\rangle\}$ by an angle ϵ .
 The probability of getting the qubit to move by ϵ increases as ϵ decreases.

$$\begin{aligned} P(|v\rangle) &= |\langle 0|v\rangle|^2 = |\cos(\epsilon)|^2 \approx 1 - \epsilon^2 & |v\rangle &= \cos(\epsilon)|0\rangle + \sin(\epsilon)|1\rangle \\ P(|w\rangle) &= |\langle 0|w\rangle|^2 = |\sin(\epsilon)|^2 \approx \epsilon^2 & |w\rangle &= -\sin(\epsilon)|0\rangle + \cos(\epsilon)|1\rangle \end{aligned} \tag{4.4}$$

So repeating this process $\approx 1/\epsilon$ times, and rotating the measurement basis by an additional angle ϵ each time, we could slowly drag the qubit from $|0\rangle$ to $|1\rangle$. *What's the likelihood that we'd ever get a measurement outcome that wasn't the one we wanted?* By the union bound, it's of order $(1/\epsilon) \times \epsilon^2 = \epsilon$, so can be made arbitrarily small. This is called **The Quantum Zeno Effect**, and one of its discoverers was Alan Turing.

Perhaps an everyday-life analog would be asking a stranger to have coffee with you, then to go dancing, etc.—there's a higher probability of success than if you just immediately ask them to marry you!

Another interesting variant of the same kind of effect is called the **Watched Pot Effect**. Say we want to keep a qubit at $|0\rangle$, but it keeps rotating towards $|1\rangle$ (it's drifting). If we keep measuring it in the $\{|0\rangle, |1\rangle\}$ basis every time the qubit has drifted by an angle ϵ , the odds of it jumping to $|1\rangle$ at any given measurement is only ϵ^2 . So if we repeat the measurements $\approx \frac{1}{\epsilon}$ times, then the probability it ending up at $|1\rangle$ is only $\approx \epsilon$, even though it would have drifted to $|1\rangle$ with certainty had we not measured.

4.4 The Elitzur-Vaidman Bomb

Another interesting phenomenon is the **Elitzur-Vaidman Bomb**, a quantum effect discovered in the early 1990's.

Say we're at a quantum airport and there's a piece of unattended luggage which could be a bomb, but opening the suitcase would trigger it. *How do we check if there's a bomb there without triggering it?*

Suppose the bomb is designed so it can accept a query made with a classical bit where $b = 0$ means we don't query and $b = 1$ means we make a query. In that case one of two things happens, we either don't make the query and learn nothing, or we make the query and risk setting off the bomb if in fact there is one. Not good!

This is the quantum airport though (IATA code BQP), so suppose instead that we can upgrade our bit to a qubit: $|b\rangle = \alpha|0\rangle + \beta|1\rangle$. We'll also assume that in the case there is no bomb, the state $|b\rangle$ gets returned to you. If there is a bomb, the bomb measures in the $\{|0\rangle, |1\rangle\}$ basis. If the outcome is $|0\rangle$, then $|0\rangle$ is returned to you, while if the outcome is $|1\rangle$, the bomb explodes.

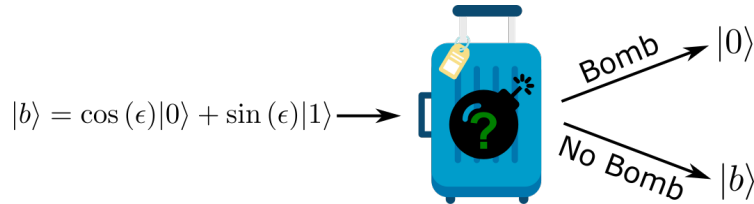


Figure 4.4: Sketch of the Elitzur-Vaidman Bomb protocol.

What we can do is start with the $|0\rangle$ state and apply the rotation

$$R_\epsilon = \begin{bmatrix} \cos(\epsilon) & -\sin(\epsilon) \\ \sin(\epsilon) & \cos(\epsilon) \end{bmatrix},$$

giving us $\cos(\epsilon)|0\rangle + \sin(\epsilon)|1\rangle$. If there's a bomb, the probability it explodes is $\sin^2(\epsilon) \approx \epsilon^2$, otherwise we get back $|0\rangle$. If there's no bomb, we get back $\cos(\epsilon)|0\rangle + \sin(\epsilon)|1\rangle$.

So repeating this process, each time applying R_ϵ , about $\frac{\pi}{2\epsilon}$ times makes the total probability of setting off the bomb (if there is a bomb) only $\frac{\pi}{2\epsilon} \sin^2(\epsilon) \approx \frac{\pi}{2}\epsilon$. Yet, by measuring our qubit to see whether it's $|0\rangle$ or $|1\rangle$, we still learn whether or not a bomb was there. If there was a bomb then by the watched pot effect the state will be $|0\rangle$ with high probability, and if there wasn't then our repeated applications of R_ϵ succeeded in rotating the state by $\frac{\pi}{2}$ and our state is $|1\rangle$. Of course, the catch is that this requires not merely a qubit on our end, but also a bomb that can be “quantumly interrogated”!

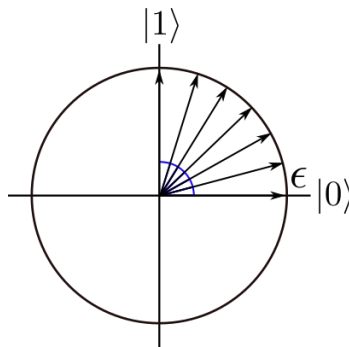


Figure 4.5: Evolution of the qubit after multiple queries with no bomb.

Lecture 5: The Coin Problem, Distinguishability, Multi-Qubit States and Entanglement

5.1 The Coin Problem

Say you have a coin, and you want to figure out if it's fair ($p = \frac{1}{2}$) or if it's biased ($p = \frac{1}{2} + \epsilon$). *How would you go about doing this?*

The classical approach to solving this problem would be to flip the coin over and over about $\frac{1}{\epsilon^2}$ times, keeping track of the number of heads and tails and appealing to the Law of Large Numbers. Standard probability stuff. This requires about $\log(\frac{1}{\epsilon^2})$ bits of memory to store the running totals. In fact, there's a theorem by Hellman and Cover from the 70s that says that any protocol to solve this problem requires at least that many bits for storage.

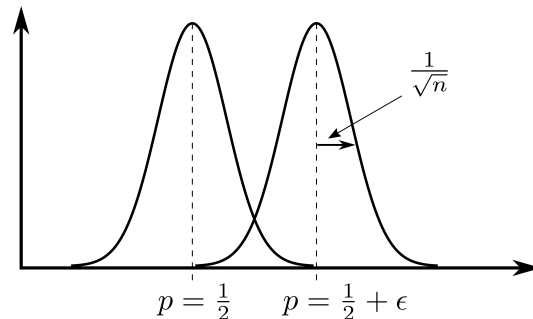


Figure 5.1: Distribution for the coin-flip probability. Since the standard error scales like $\frac{1}{\sqrt{n}}$, where n is the number of coin flips, if we want to distinguish reliably between two distributions whose means are separated by ϵ we need $\frac{1}{\sqrt{n}} \sim \epsilon \rightarrow n \sim \frac{1}{\epsilon^2}$ coin flips.

What if instead we used quantum states? We can start with a qubit in

the $|0\rangle$ state, and consider the two rotations R_ϵ and $R_{-\epsilon}$, which rotate by ϵ and $-\epsilon$ radians respectively. We can repeatedly flip the coin, and if it lands tails apply R_ϵ (rotating clockwise) and if it lands heads apply $R_{-\epsilon}$ (rotating counterclockwise). After many flips ($\sim \frac{1}{\epsilon^2}$) we can then measure the qubit and statistically infer that if it's in the $|0\rangle$ state the coin was most likely fair, while if it's in the $|1\rangle$ state the coin is most likely biased. You might raise a few objections about the protocol:

- ▶ *Won't counting out the right number of steps again require a lot of storage?*
 - No. We can give a protocol with a half-life (some independent probability of halting at each step) causing it to repeat approximately the number of times we want it to.
- ▶ *What about if the qubit drifts by a multiple of π ? Won't that make a biased coin look fair?*
 - That's possible, but we can make it so that a biased coin is more likely to land on $|1\rangle$ than a fair coin.

This is our first example of a quantum protocol getting a resource advantage: the quantum solution uses *1 qubit of storage* as opposed to the classical solution's $\log(\frac{1}{\epsilon^2})$ bits.

This result was shown by Professor Aaronson and his former student Andy Drucker. It wasn't a particularly hard problem, but no one had asked the question before. There's still "low hanging fruit," even in the mechanics of a single qubit!

5.2 Distinguishability of Quantum States

Given two orthogonal quantum states $|v\rangle$ and $|w\rangle$, there's a basis we can measure in which distinguishes them with certainty. Given a pair of states which are co-linear (like $|v\rangle$ and $-|v\rangle$) however, they are indistinguishable.

This points us towards the magnitude of the inner product $|\langle v|w\rangle|$ as a good measure of the distinguishability for arbitrary pure states. Suppose we're given states like those in Figure 5.4 and we want to know: What measurement would minimize the chance of making a mistake in differentiating $|v\rangle$ from $|w\rangle$, assuming that being given either is equally likely?

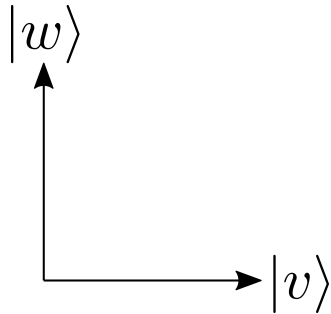


Figure 5.2: Orthogonal states



Figure 5.3: Co-linear states

You may want to measure in the $|v\rangle, |v^\perp\rangle$ basis, as it would eliminate one kind of error completely (not getting $|v\rangle$ ensures the state was $|w\rangle$). But if you just want to maximize the probability of getting the right answer, and if $|v\rangle$ and $|w\rangle$ are equally likely, then there's a better way, illustrated in Figure 5.5. Take the bisector of $|v\rangle$ and $|w\rangle$ and define the measurement basis by using the states 45° to either side. When we perform the measurement in this basis we output the original vector closest to the measurement result as the outcome.

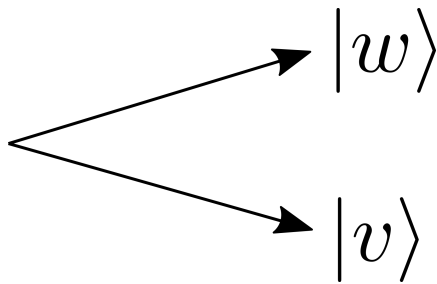


Figure 5.4: Nonorthogonal states $|v\rangle$ and $|w\rangle$.

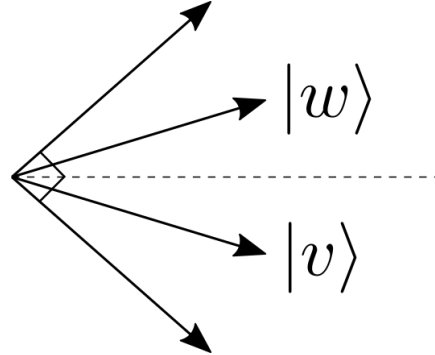


Figure 5.5: Optimal measurement basis for minimizing the error of distinguishing nonorthogonal states $|v\rangle$ and $|w\rangle$.

5.3 Multi-Qubit States and Operations

A general state of two qubits is:

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$$

where the probabilities of the four outcomes are given by the Born rule, as with a single qubit:

$$\begin{aligned} P(|00\rangle) &= |\alpha|^2 & P(|01\rangle) &= |\beta|^2 \\ P(|10\rangle) &= |\gamma|^2 & P(|11\rangle) &= |\delta|^2 \end{aligned}$$

In principle there's no distance limitation between entangled qubits. One qubit could be with you on Earth, and the other could be with your friend on the moon. In such a case, though, you'd only be able to measure the first qubit. The probability of getting $|0\rangle$ is $|\alpha|^2 + |\beta|^2$ because those are the amplitudes compatible with the first qubit being $|0\rangle$. The probability of getting $|1\rangle$ is $|\gamma|^2 + |\delta|^2$.

Suppose I measure the first qubit and get the outcome $|0\rangle$. *What can I say about the second qubit?*

Well, we've narrowed down the possibilities for the joint state to $|00\rangle$ and $|01\rangle$. The state is thus now in the superposition

$$|0\rangle \otimes \frac{\alpha|0\rangle + \beta|1\rangle}{\sqrt{|\alpha|^2 + |\beta|^2}}, \quad (5.1)$$

where the factor of $\sqrt{|\alpha|^2 + |\beta|^2}$ in the denominator ensures that the result is properly normalized. This is called the **Partial Measurement Rule**. This is actually the last "basic rule" of quantum mechanics that we'll see in the course; everything else is just a logical consequence of rules we've already covered.

5.3.1 Multi-Qubit Operations

One of the most common 2-qubit operations we'll encounter is one we've already seen: the CNOT gate, which flips the second bit if and only if the first bit is 1. Recall that the matrix corresponding to this operation is given by

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \quad (5.2)$$

What if instead we wanted to apply an operation where we do nothing to the first qubit and apply a NOT to the second qubit? The matrix corresponding to this operation is

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \quad (5.3)$$

which can be written in tensor product notation as

$$I \otimes \text{NOT} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad (5.4)$$

Likewise, if we want to apply a NOT to the first qubit and do nothing to the second qubit we can apply $\text{NOT} \otimes I$, which in matrix representation is

$$\text{NOT} \otimes I = \begin{array}{c} \begin{matrix} 00 & 01 & 10 & 11 \end{matrix} \\ \begin{matrix} 00 \\ 01 \\ 10 \\ 11 \end{matrix} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{array} \quad (5.5)$$

Remember that rows represent input amplitudes and columns represent output amplitudes, so for $\text{NOT} \otimes I$ the amplitude on 00 in the input is the amplitude on 10 in the output.

Very often in quantum information we'll want to take a group of qubits and perform an operation on one of them: say, "Hadamard the third qubit." What that really means is applying the unitary matrix $I \otimes I \otimes H \otimes I \otimes \dots \otimes I$. The desired operation on the relevant qubit(s) is tensor-producted with the identity operation on all the other qubits.

What's $H \otimes H$?

$$\frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \quad (5.6)$$

Why should it look like this? Let's look at the first row: $H \otimes H |00\rangle = |++\rangle$. For the second row, $H \otimes H |01\rangle = |+-\rangle$, and so on. All of the two-qubit unitaries we've seen were built up using tensor products of single-qubit unitaries, *except* for the CNOT, where the first qubit affects the second. We'll need operations like CNOT in order to have one qubit affect another.

5.3.2 Entanglement

Let's see the multi-qubit operations of Section 5.3.1 in action by calculating the result of applying the circuit below. The sequence of operations in Figure 5.6 and their effects on the input are given in ket notation in Equation 5.7 and in vector notation in Equation 5.8.

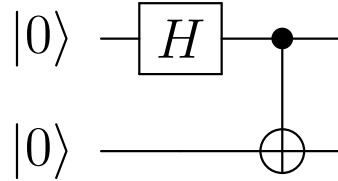


Figure 5.6: Circuit for producing a Bell pair.

$$|00\rangle \rightarrow |+\rangle \otimes |0\rangle = \frac{|00\rangle + |10\rangle}{\sqrt{2}} \rightarrow \frac{|00\rangle + |11\rangle}{\sqrt{2}} \quad (5.7)$$

$$(\text{CNOT})(H \otimes I) \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \rightarrow \text{CNOT} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ \frac{1}{\sqrt{2}} \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} \quad (5.8)$$

The action of the CNOT can also be written as $|x, y\rangle \rightarrow |x, y \oplus x\rangle$. The state that this circuit ends on, $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$, is called the **Singlet** or the **Bell Pair** or the **EPR Pair**. This state is particularly interesting because measuring the first qubit collapses the state of the second qubit. The state can't be factored into a tensor product of the first qubit's state and the second qubit's state. Such a state is called *entangled*, which for pure states simply means: not decomposable into a tensor product.

A state that's not entangled is called unentangled or separable or a product state (for pure states, which are the only kind being discussed at this point, all three of these mean the same thing).

The basic rules of quantum mechanics, which we saw earlier, force entanglement to exist. It was noticed quite early in the history of the field. It turns out that most states are entangled.

As we mentioned earlier, entanglement was arguably what troubled Einstein the most about quantum mechanics. He thought that it meant that quantum mechanics must entail "spooky action at a distance." That's because, while typically particles need to be close to become entangled, once they're entangled you can separate them to an arbitrary distance and they'll

stay entangled (assuming nothing else is done to them). This has actually been demonstrated experimentally for distances of up to 150 miles (improved to a couple thousand miles by Chinese satellite experiments, while this course was being taught!).

Let's say that Alice and Bob entangle a pair of particles by setting their state to $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$. Then Alice brings her particle to the moon while Bob stays on Earth. If Alice measures her particle, she can *instantaneously* know whether Bob will observe a $|0\rangle$ or a $|1\rangle$ when he measures his.

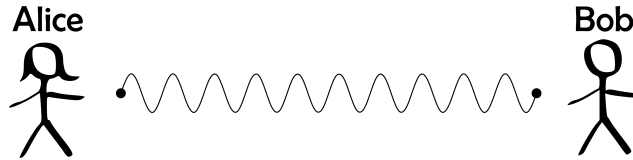


Figure 5.7: We often denote shared entanglement between two parties with a squiggly line (you know, cause entanglement is “spooky” and “weird”).

This bothered Einstein, but others thought that it wasn't that big a deal. After all, Alice doesn't get to control the outcome of her measurement! She sees $|0\rangle$ and $|1\rangle$ with equal probability, which means that in this case, the “spooky action” can be explained as just a correlation between two random variables, as we could already see in the classical world. However, a famous 1935 paper of Einstein, Podolsky, and Rosen brought up a further problem: namely, there are other things Alice could do instead of measuring in the $\{|0\rangle, |1\rangle\}$ basis.

What happens if Alice measures in the $\{|+\rangle, |-\rangle\}$ basis? She'll get either $|+\rangle$ or $|-\rangle$, as you might expect. Indeed, we can model the situation by Alice Hadamarding her qubit and then measuring in the $\{|0\rangle, |1\rangle\}$ basis. Alice Hadamarding gives us the state

$$(H \otimes I) \left(\frac{|00\rangle + |11\rangle}{2} \right) = \frac{|00\rangle + |01\rangle + |10\rangle - |11\rangle}{2}.$$

So now, applying the partial measurement rule what is Bob's state? If Alice sees $|0\rangle$, then Bob's qubit collapses to

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

Conversely, if Alice sees $|1\rangle$ then Bob's qubit collapses to

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

Einstein, Podolsky and Rosen went on to talk about how this is more troubling than before. If Alice measures in the $\{|0\rangle, |1\rangle\}$ basis, then Bob's state collapses to $|0\rangle$ or $|1\rangle$, but if she measures in the $\{|+\rangle, |-\rangle\}$ basis, then Bob's state collapses to $|+\rangle$ or $|-\rangle$. And that looks a lot like faster-than-light communication!

How can we explain this? One thing we can do is ask “what happens if Bob makes a measurement?”

- ▶ In the case where Alice measured her qubit in the $\{|0\rangle, |1\rangle\}$ basis, Bob will see $|0\rangle$ or $|1\rangle$ with equal probability if he measures his qubit in the same basis.
- ▶ In the case where Alice measured her qubit in the $\{|+\rangle, |-\rangle\}$ basis, Bob will still see $|0\rangle$ or $|1\rangle$ with equal probability if he measures his qubit in the $\{|0\rangle, |1\rangle\}$ basis (as an exercise, check this).

So, at least in this case, the probability that Bob sees $|0\rangle$ or $|1\rangle$ is the same regardless of what Alice chooses to do. So, it looks like there might be something more general going on here! In particular, a different description should exist of Bob's part of the state that's totally unaffected by Alice's measurements—thereby making manifest the principle of no faster-than-light communication. Which brings us to the next lecture...

Lecture 6: Mixed States

So far we've only talked about **pure** states (i.e., isolated superpositions), but you can also have quantum superposition layered together with regular, old probabilistic uncertainty. This becomes extremely important when we talk about states where we're only measuring one part. Last time we discussed the **Bell Pair**, and how if Alice measures her qubit in any basis, the state of Bob's qubit collapses to whichever state she got for her qubit. Even so, there's a formalism that helps us see why Bob can't do anything to learn which basis Alice makes her measurement in, and more generally, why Alice can't transmit *any* information instantaneously—in keeping with special relativity. This is the formalism of . . .

6.1 Mixed States

Mixed states in some sense are just probability distributions over quantum superpositions. We can define a mixed state as a distribution over quantum states, $\{p_i, |\psi_i\rangle\}$, meaning that with probability p_i the state is $|\psi_i\rangle$.

Note that the $|\psi_i\rangle$'s don't have to be orthogonal

Thus, we can think of a pure state as a degenerate case of a mixed state where all the probabilities are 0 or 1. The tricky thing about mixed states is that *different probability distributions over pure states, can give rise to exactly the same mixed state* (we'll see an example shortly). But to make manifest why information doesn't travel faster than light, we need a representation for mixed states that's unique. This representation is called **Density Matrices**.

6.1.1 Density Matrices

The density matrix representation of a mixed state $\{p_i, |\psi_i\rangle\}$ is given by

$$\rho = \sum_i p_i |\psi_i\rangle \langle\psi_i| \quad (6.1)$$

where $|\psi_i\rangle \langle\psi_i|$ denotes the **outer product** of $|\psi\rangle$ with itself. The outer product is the matrix which you get by multiplying

$$\begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{N-1} \end{bmatrix} [\alpha_0^* \quad \alpha_1^* \quad \cdots \quad \alpha_{N-1}^*] = \begin{bmatrix} |\alpha_0|^2 & & & \\ & \ddots & & \\ & & \alpha_i \alpha_j^* & \\ & & \alpha_j \alpha_i^* & \ddots \\ & & & & |\alpha_{N-1}|^2 \end{bmatrix}. \quad (6.2)$$

Note that $\alpha_i \alpha_j^* = (\alpha_j^* \alpha_i)^*$, which means that the matrix is its own conjugate transpose $\rho = \rho^\dagger$. This makes ρ a **Hermitian Matrix**. For the standard basis states, for example, we get

$$|0\rangle \langle 0| = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \quad \text{and} \quad |1\rangle \langle 1| = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

Therefore, an even mixture of them would be

$$\frac{|0\rangle \langle 0| + |1\rangle \langle 1|}{2} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} = \frac{I}{2}. \quad (6.3)$$

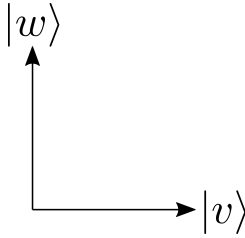
Similarly,

$$|+\rangle \langle +| = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \quad |-\rangle \langle -| = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

and

$$\frac{|+\rangle \langle +| + |-\rangle \langle -|}{2} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} = \frac{I}{2}. \quad (6.4)$$

Notice that an equal mixture of $|0\rangle$ and $|1\rangle$ is different from an equal superposition of $|0\rangle$ and $|1\rangle$ (a.k.a. $|+\rangle$), and so they have different density matrices. However, the mixture of $|0\rangle$ and $|1\rangle$ and the mixture of $|+\rangle$ and $|-\rangle$ have the same density matrix, which makes sense because Alice converting between the two bases in our Bell pair example should maintain Bob's density matrix representation of his state.



In fact, this is true of whichever basis Alice chooses: for any two orthogonal vectors $|v\rangle$ and $|w\rangle$, we have that

$$\frac{|v\rangle\langle v| + |w\rangle\langle w|}{2} = \frac{I}{2}. \quad (6.5)$$

Measuring ρ in the basis $\{|0\rangle \dots |N-1\rangle\}$ gives us outcome $|i\rangle$ with probability $P(|i\rangle) = \rho_{ii} = \langle i|\rho|i\rangle$. So the diagonal entries of the density matrix directly represent probabilities.

You don't need to square them or anything because the Born rule is already encoded in the density matrix (i.e. $\alpha_i\alpha_i^ = |\alpha_i|^2$)*

In particular, a density matrix that's diagonal is just a fancy way of writing a classical probability distribution.

$$\begin{bmatrix} p_0 & & 0 \\ & \ddots & \\ 0 & & p_{N-1} \end{bmatrix} \quad (6.6)$$

A pure state written as a density matrix, for example $|+\rangle\langle +|$, would look like $\begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix}$, a matrix of rank one. Indeed, a density matrix has rank 1 *if and only if* it represents a pure state.

What if we want to measure a density matrix in a different basis? Measuring ρ in the basis $\{|v\rangle, |w\rangle\}$ will give $P(|v\rangle) = \langle v|\rho|v\rangle$ and $P(|w\rangle) = \langle w|\rho|w\rangle$.

You can think of a density matrix as encoding not just one but infinitely many probability distributions, because you can measure in any basis.

The matrix $I/2$ that we've encountered above as the even mixture of $|0\rangle$ and $|1\rangle$ (and also of $|+\rangle$ and $|-\rangle$) is called the **Maximally Mixed State**. This state is basically just the outcome of a classical coin flip, and it has a special property: regardless of the basis we measure it in, both outcomes will be equally likely. So for every basis $\{|v\rangle, |w\rangle\}$ we get the probabilities

$$\begin{aligned} \langle v|\frac{I}{2}|v\rangle &= \frac{1}{2}\langle v|v\rangle = \frac{1}{2}, \\ \langle w|\frac{I}{2}|w\rangle &= \frac{1}{2}\langle w|w\rangle = \frac{1}{2}. \end{aligned}$$

This explains why Alice, no matter what she tries, is unsuccessful in sending a message to Bob by measuring her half of a Bell pair. Namely, because the maximally mixed state in any other basis is *still the maximally mixed state*. The generalization of this fact to any state shared by Alice and Bob and to any operation performed by Alice is called the **No-Communication Theorem**.

So how do we handle unitary transformations with density matrices? Since $\rho = \sum_i p_i |\psi_i\rangle \langle \psi_i|$, applying U to ρ means that ρ gets mapped to

$$\sum_i p_i (U |\psi_i\rangle) (U |\psi_i\rangle)^\dagger = \sum_i p_i U |\psi_i\rangle \langle \psi_i| U^\dagger = U \left(\sum_i p_i |\psi_i\rangle \langle \psi_i| \right) U^\dagger = U \rho U^\dagger. \quad (6.7)$$

You can pull out the U 's since it's the same one applied to each state in the mixture.

It's worth noting that getting n^2 numbers in the density matrix isn't some formal artifact; we really do need all those extra parameters. *What do the off-diagonal entries represent?*

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

The off-diagonal entries are where the “quantumness” of the state resides. They're where the potential interference between $|0\rangle$ and $|1\rangle$ is represented. The off-diagonal entries can vary depending on relative phase: $|+\rangle \langle +|$ has positive off-diagonal entries, $|-\rangle \langle -|$ has negative off-diagonal entries and $|i\rangle \langle i| = \begin{bmatrix} \frac{1}{2} & \frac{-i}{2} \\ \frac{i}{2} & \frac{1}{2} \end{bmatrix}$ has off-diagonal entries of opposite signs. Later we'll see that as a quantum system interacts with the environment, the off-diagonal entries tend to get pushed down toward 0.

The density matrices in experimental quantum information papers typically look like $\begin{bmatrix} \frac{1}{2} & \epsilon \\ \epsilon^ & \frac{1}{2} \end{bmatrix}$. The bigger the off-diagonal values, the better the experiment, because it represents them seeing more of the quantum effect! A caveat though: off-diagonal entries are basis-dependent. In fact, as we'll see, they can always be made 0 by a suitable change of basis.*

6.1.2 Properties of Density Matrices

Which matrices can arise as density matrices? We're effectively asking: what constraints does the equation $\rho = \sum_i p_i |\psi_i\rangle \langle \psi_i|$ put on the matrix ρ ? Well, such a ρ must be:

- ▶ Square
- ▶ Hermitian
- ▶ Trace 1 (which is to say $\sum_i \rho_{ii} = 1$)

Could $M = \begin{bmatrix} \frac{1}{2} & -10 \\ -10 & \frac{1}{2} \end{bmatrix}$ be a density matrix?

No! Measuring this in the $\{|+\rangle, |-\rangle\}$ basis would give $\langle +|M|+\rangle = 19/2$. Bad!

Remember that you can always transform ρ to $U\rho U^\dagger$, whose diagonal then has to be a probability distribution. If we want that condition to hold for all U , then we need to add the restriction:

- ▶ All eigenvalues are non-negative (ρ is **positive-semidefinite** or **PSD**)

As a refresher, for the matrix ρ , the eigenvectors $|x\rangle$ are the vectors that satisfy the equation $\rho|x\rangle = \lambda|x\rangle$ for some eigenvalue λ . If we had an eigenvector $|x\rangle$ with a negative eigenvalue then the probability $\langle x|\rho|x\rangle = \lambda$ would be negative, which is nonsense.

Could we have missed a condition? Let's check. *We claim: any Hermitian PSD matrix with trace 1 can arise as a density matrix of a quantum state.*

For such a ρ , we can represent it in the form $\rho = \sum_i \lambda_i |\psi_i\rangle \langle \psi_i|$ where the $|\psi_i\rangle$ are the (normalized) eigenvectors of ρ . Then $\langle \psi_i|\rho|\psi_i\rangle = \lambda_i$, so the λ_i 's sum to $\text{Tr}(\rho) = 1$. This process of obtaining eigenvalues and eigenvectors is called **eigendecomposition**. We know the eigenvalues will be real because the matrix is Hermitian and they're non-negative because the matrix is PSD. For every density matrix ρ , there's a U such that $U\rho U^\dagger$ is diagonal (with ρ 's eigenvalues along its diagonal). Namely, the U that switches between the standard basis and ρ 's eigenbasis.

One important quantity you can always compute for density matrices is the **rank** defined as

$$\text{rank}(\rho) = \text{The number of non-zero eigenvalues } \lambda \text{ (with multiplicity)}. \quad (6.8)$$

A density matrix of rank n might look, for example, like that in Equation 6.6, while a density matrix of rank 1 represents a pure state.

We know from linear algebra that the rank of an $n \times n$ matrix is always at most n . Physically, this means that every n -dimensional mixed state can be written as a mixture of at most n pure states.

In general, rank tells you the minimum number of pure states that you have to mix to reach a given mixed state.

6.1.3 Partial Trace and Reduced Density Matrices

Now, consider the 2-qubit pure state

$$\frac{|00\rangle + |01\rangle + |10\rangle}{\sqrt{3}}. \quad (6.9)$$

We'll give the first qubit to Alice and the second to Bob. *How does Bob calculate the density matrix corresponding to his system, also called the **Reduced Density Matrix** or **Local Density Matrix**?* Start by picking some orthogonal basis for Alice's side. The state can be rewritten as

$$\sqrt{\frac{2}{3}}|0\rangle|+\rangle + \sqrt{\frac{1}{3}}|1\rangle|0\rangle,$$

which lets you calculate Bob's density matrix as

$$\frac{2}{3}|+\rangle\langle+| + \frac{1}{3}|0\rangle\langle 0| = \frac{2}{3} \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} + \frac{1}{3} \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} \end{bmatrix}$$

In general, if you have a bipartite pure state, it'll look like $|\psi\rangle = \sum_{i,j=0}^{N-1} \alpha_{i,j} |i\rangle |j\rangle$ and Bob's reduced density matrix can be obtained using

$$(\rho^B)_{j,j'} = \sum_i \alpha_{i,j} \alpha_{i,j'}^*. \quad (6.10)$$

The process of going from a pure state of a composite system, to the mixed state of part of the system, is called **tracing out**.

The key points:

- ▶ A density matrix encodes all of the physically observable information about a quantum system.
 - Two quantum states will lead to different probabilities for some measurement *iff* they have different density matrices.
- ▶ No-Communication Theorem

- If Alice and Bob share an entangled state, nothing Alice chooses to do will have any effect on Bob's reduced density matrix.

In other words, there's no observable effect on Bob's end. Which is the fundamental reason why quantum mechanics is compatible with the limitations of relativity.

We've already seen particular examples of both statements. But both of them hold in full generality, and you'll prove that in your homework!

OK, just to get you started a bit: recall that the No-Communication Theorem says that, if Alice and Bob share an entangled state

$$|\psi\rangle = \sum_{i,j=0}^{N-1} \alpha_{i,j} |i\rangle_{\text{Alice}} |j\rangle_{\text{Bob}},$$

there's nothing that Alice can do to her subsystem that affects Bob's reduced density matrix. You already have the tools to prove this: just calculate Bob's reduced density matrix, then apply a unitary transformation to Alice's side, then see if Bob's density matrix changes. Or have Alice measure her side, and see if Bob's reduced density matrix changes.

Note that if we condition on the outcome of Alice's measurement, then we do need to update Bob's local density matrix to reflect the new knowledge: if Alice sees i then Bob sees j , etc... But that's not terribly surprising, since the same would also be true even with classical correlation! In particular, this doesn't provide a mechanism for faster-than-light communication.

To review, we've seen three different types of states in play, each more general than the last:

- ▶ Basis States or Classical States
 - Individual states in some computational basis e.g. $|i\rangle$.
- ▶ Pure States
 - Superpositions of basis states $|\psi\rangle = \sum_i \alpha_i |i\rangle$
- ▶ Mixed States
 - Classical probability distributions over pure states $\rho = \sum_i p_i |\psi_i\rangle \langle \psi_i|$.

Which represents the actual physical reality: pure or mixed states? It's complicated. Sometimes we use density matrices to represent our probabilistic ignorance of a pure state. But when we look at part of an entangled state, a mixed state is the most complete representation possible that only talks about the part that we're looking at. We'll generally just focus on what these representations are useful for.

Lecture 7: The Bloch Sphere, No-Cloning Theorem and Wiesner's Quantum Money Scheme

7.1 The Bloch Sphere

The **Bloch Sphere** is a geometric representation of the set of all possible mixed states of a qubit. We've often drawn the state of a qubit as a point on the unit circle, which is already a little awkward: half of the circle is going to waste since $|\psi\rangle$ and $-|\psi\rangle$ both represent the same physical state (they have the same density matrix).

Instead, what if we chose a representation where vectors that pointed in *opposite* directions were orthogonal? With this choice of convention we get the Bloch sphere, as seen in Figure 7.1.

We can see that $|+\rangle$ and $|-\rangle$ should be between $|0\rangle$ and $|1\rangle$. Then we can add $|i\rangle$ and $|-i\rangle$ as a third dimension. In this representation points on the surface of the sphere are pure states, such that if they're 180° apart they're orthogonal.

What about mixed states? Well we know that the maximally mixed state, $\frac{I}{2}$, can be defined as

$$\frac{|0\rangle\langle 0| + |1\rangle\langle 1|}{2}, \quad \frac{|+\rangle\langle +| + |-\rangle\langle -|}{2}, \quad \text{or} \quad \frac{|i\rangle\langle i| + |-i\rangle\langle -i|}{2}.$$

More generally we can define the maximally mixed state as $\frac{|v\rangle\langle v| + |v^\perp\rangle\langle v^\perp|}{2}$ for any orthogonal pair of states $|v\rangle$ and $|v^\perp\rangle$. The sum of any two of these vectors on the sphere is the origin.

More generally, we can in this way represent any mixed state as a point inside of the sphere. A mixture of any states $|v\rangle$ and $|w\rangle$, represented as points on the surface of the sphere, will be a point on the line segment connecting

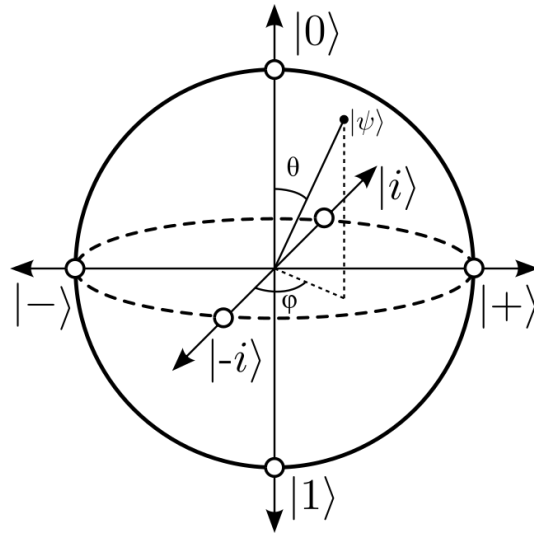


Figure 7.1: The Bloch sphere representation of a qubit state. Antipodal points on the surface of the sphere correspond to orthogonal states. Pure states live on the surface of the sphere while mixed states correspond to points in the interior. The center of the sphere is a special point and corresponds to the maximally mixed state $\frac{I}{2}$.

the two.

We can show geometrically that every 1-qubit mixed state can be written as a mixture of only two pure states. *Why?* Because you can always draw a line that connects any pure state you want to some point in the sphere representing a mixed state, and then see which other pure state the line intersects on its way out. The point can then be described as some convex combination of the vectors representing the pure states. This is visually represented in Figure 7.2.

Experimentalists love the Bloch sphere, because it works identically to how spin works with electrons and other spin- $\frac{1}{2}$ particles. You can measure the particle’s “spin” relative to any axis of the sphere, and the result will be that the electron is spinning either clockwise or counterclockwise relative to the axis. The particle’s spin state is literally a qubit, which collapses to one of the two possible spin states on measurement.

The weird part about spin- $\frac{1}{2}$ particles is that you *could* have asked the direction of the spin relative to any other axis and still would have gotten that it was either clockwise or counter-clockwise relative to that axis. So what’s really going on: *what’s the real spin direction?* Well, the actual state is just some point on the Bloch sphere, so there is a “real spin direction,” but there’s also no measurement that reliably tells us that direction. The crazy part here

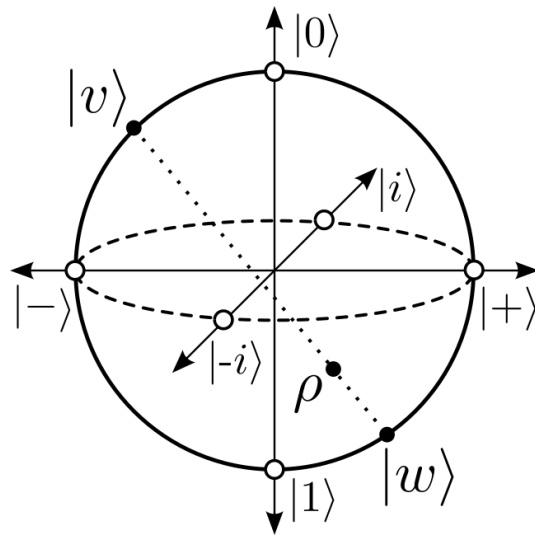


Figure 7.2: A mixed state of a qubit can always be represented by a convex combination of pure states on the surface of the Bloch sphere. To get one such (there are infinitely many) representation, pick some state on the surface $|v\rangle$ and draw a line through $|v\rangle$ and the point representing the state ρ . The other point on the surface gives $|w\rangle$, the other state in the mixture. The relative lengths of the line segments between $|v\rangle$ and ρ and between $|w\rangle$ and ρ give the weights of the mixture.

is how the three-dimensionality of the Bloch sphere perfectly syncs up with the three-dimensionality of actual physical space.

7.1.1 Quantum Gates in the Bloch Sphere Representation

It's often useful to visualize the effect of quantum gates geometrically by mapping their behavior to corresponding changes on the Bloch sphere. In the Bloch sphere representation, every quantum gate can be described as a 3D rotation by some angle θ about some axis defined by the eigenvectors of the gate. Some examples of this can be seen in Figure 7.3.

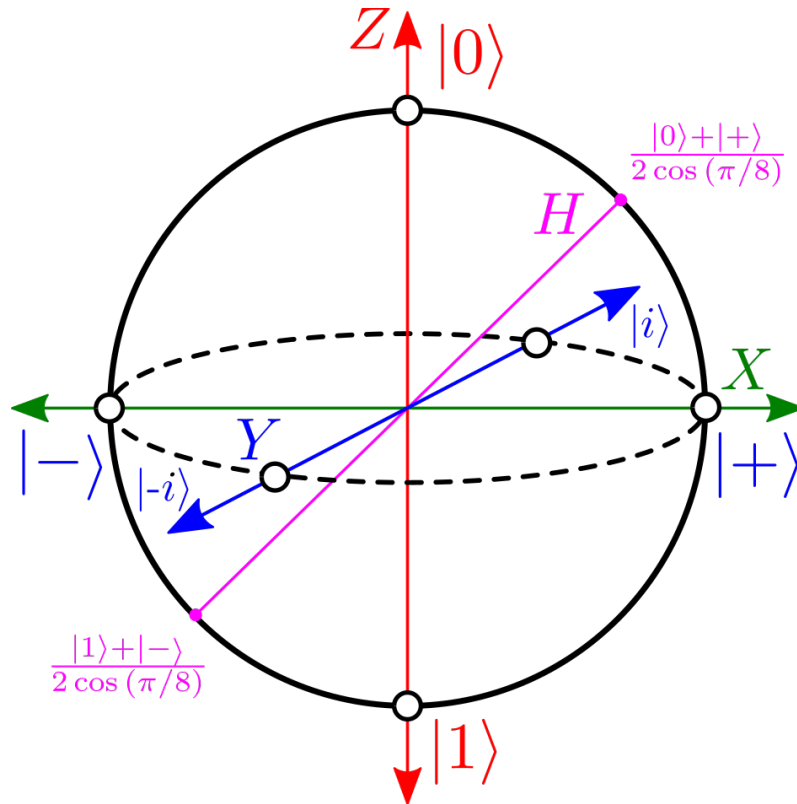


Figure 7.3: Applying gates X , Y , Z or H is the same as doing a half-turn about their respective axes. S corresponds to a quarter turn around the Z -axis (in the $|+\rangle$ to $|1\rangle$ direction). $T^2 = S$, so T corresponds to an eighth turn around the Z -axis. $R_{\pi/4}$ corresponds to a quarter turn about the Y -axis.

7.2 The No-Cloning Theorem

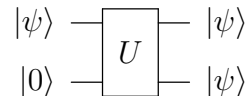
We've seen how entanglement seems to lead to "non-local effects," like for the state $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$, where if Alice measures her qubit then she learns the state of Bob's. The reason that Alice isn't communicating faster than light boils down to Bob not being able to tell if his qubit's state is in the $\{|0\rangle, |1\rangle\}$ basis or the $\{|+\rangle, |-\rangle\}$ basis. *But what if Bob could make unlimited copies of his qubit?* He could figure out which state he had through repeated measurements, and so he'd be able to tell what basis Alice measured in. Faster than light communication!

*Learning a classical description of a quantum state, given lots of copies of the state, is called **Quantum State Tomography**.*

It turns out that we can prove that a procedure to reliably copy an unknown quantum state cannot exist. It's easy to prove, but it's a fundamental fact about quantum mechanics. In fact, we already saw one proof: namely, cloning would imply superluminal communication, which would violate the No-Communication theorem that you proved in the homework! But let's see more directly why cloning is impossible.

Let's try to clone a single qubit, $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$.

In our quantum circuit we want to apply some unitary transformation that takes $|\psi\rangle$ and an ancilla as input, and produces two copies of $|\psi\rangle$ as output.



Algebraically our cloner U would need to implement the transformation

$$\begin{aligned} (\alpha|0\rangle + \beta|1\rangle) \otimes |0\rangle &\rightarrow (\alpha|0\rangle + \beta|1\rangle) \otimes (\alpha|0\rangle + \beta|1\rangle) \\ &= \alpha^2|00\rangle + \alpha\beta|01\rangle + \alpha\beta|10\rangle + \beta^2|11\rangle. \end{aligned} \quad (7.1)$$

In matrix notation this looks like

$$\begin{bmatrix} \alpha^2 \\ \alpha\beta \\ \alpha\beta \\ \beta^2 \end{bmatrix} = U \begin{bmatrix} \alpha \\ 0 \\ \beta \\ 0 \end{bmatrix}. \quad (7.2)$$

The problem? This transformation isn't linear (it has quadratic terms), so it can't be unitary!

To clarify, a known procedure that outputs some state $|\psi\rangle$ can be rerun to get many copies of $|\psi\rangle$. What the No-Cloning Theorem says is that if $|\psi\rangle$ is given to you but is otherwise unknown then you can't make a copy of it.

Another clarification: CNOT seems like a copying gate—as it maps $|00\rangle \rightarrow |00\rangle$ and $|10\rangle \rightarrow |11\rangle$. *So why doesn't it violate the No-Cloning Theorem?* Because it only copies if the input state is $|0\rangle$ or $|1\rangle$. Classical information *can* be copied. Doing CNOT on $|+\rangle|0\rangle$ produces the Bell pair $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$; this sort of copies the first qubit in an entangled way, but that's different than making a copy of $|+\rangle$. Having two qubits in the local state $\frac{I}{2}$ is not the same as having two in the state $|+\rangle$. In general, for any orthonormal basis you can clone the basis vectors, if you know that your input state is one of them.

Since the No-Cloning Theorem is so important, we'll present another proof of it. A unitary transformation can be defined as a linear transformation that preserves inner product. Which is to say that the angle between $|v\rangle$ and $|w\rangle$ is the same as the angle between $U|v\rangle$ and $U|w\rangle$. Thus $\langle v|U^\dagger U|w\rangle = \langle v|w\rangle$.

What would a cloning map do to this inner product? Let $|\langle v|w\rangle| = c$; then

$$|(\langle v| \otimes \langle v|)(|w\rangle \otimes |w\rangle)| = |\langle v|w\rangle \langle v|w\rangle| = c^2.$$

Now, c only ever equals c^2 if $c = 0$ or $c = 1$, so the transformation can only copy perfectly if $|v\rangle$ and $|w\rangle$ belong to the same orthonormal basis.

There's a fact in classical probability that provides a nice analog to the No-Cloning Theorem. If we're given the outcome of a coin flip—from a coin that lands heads with some unknown probability p —can we simulate a second independent flip of the same coin without having access to the coin? To do so we'd need a stochastic S matrix whose effect is

$$\begin{bmatrix} p^2 \\ p(1-p) \\ p(1-p) \\ (1-p)^2 \end{bmatrix} = S \begin{bmatrix} p \\ 0 \\ 1-p \\ 0 \end{bmatrix}$$

but once again the transformation we'd need isn't even linear, let alone stochastic.

The No-Cloning Theorem has all sorts of applications to science fiction, because you can't make arbitrary copies of a physical system (say for teleporting yourself) if any of the relevant information (say, in your brain) were encoded

in quantum states that didn't belong to a known orthogonal basis.

7.3 Quantum Money

Quantum Money is a striking application of the No-Cloning Theorem. In some sense it was the first idea in quantum information and was involved in the birth of the field. The original quantum money scheme was proposed by Wiesner in 1969, though it was only published in the 80's.

Wiesner had left research by then and had chosen to become a manual laborer.

Wiesner realized that the quantum No-Cloning Theorem—though it wasn't yet called that—could be useful to prevent counterfeiting of money. In practice, mints use special ink, watermarks, etc. . . , but all such devices basically just lead to an arms race with the counterfeiters. So Wiesner proposed using qubits to create money that would be physically impossible to counterfeit. The immediate problem is that a money scheme needs not only unclonability but also verifiability—that is, you need to be able to check whether a bill is genuine. *How did Wiesner solve this problem?*

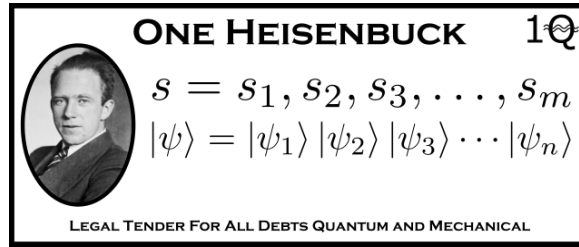
7.3.1 Wiesner's Quantum Money Scheme

The bank prints quantum bills (we'll assume for simplicity that they're all same denomination). Each bill has:

- ▶ A classical serial number $s \in \{0, 1\}^m$
- ▶ A quantum state $|\psi_{f(s)}\rangle$
 - The qubits in this state are unentangled, and each will always be in one of four states: $|\psi_{00}\rangle = |0\rangle$, $|\psi_{01}\rangle = |1\rangle$, $|\psi_{10}\rangle = |+\rangle$, or $|\psi_{11}\rangle = |-\rangle$.

The bank maintains a giant database that stores for each bill in circulation the classical serial number s as well as a string $f(s)$ that encodes what the quantum state attached to bill is supposed to be.

Wiesner's scheme, like all quantum money schemes, has an important practical problem though: you need to ensure



that the qubits in a bill don't lose their state (coherence). With current technology, qubits in a lab decohere in anywhere from nanoseconds to an hour. Qubits stored in a wallet would decohere much faster!

To verify a bill, you bring it back to the bank. The bank verifies the bill by looking at the serial number, and then measuring each qubit in the bill in the basis in which it was supposed to be prepared. That is, if the qubit was supposed to be $|0\rangle$ or $|1\rangle$, then measure in the $\{|0\rangle, |1\rangle\}$ basis; if it was supposed to be $|+\rangle$ or $|-\rangle$, then measure in the $\{|+\rangle, |-\rangle\}$ basis. For each measurement, check that you get the expected outcome.

Consider a counterfeiter who doesn't know which basis each qubit is supposed to be in, so they guess the bases uniformly at random. They only have a $(\frac{1}{2})^n$ chance of making all n guesses correctly. Of course one could imagine a more sophisticated counterfeiter, but it's possible to prove that regardless of what the counterfeiter does, if they map a single input bill to two output bills then the output bills will both pass verification with probability at most $(\frac{3}{4})^n$.

Wiesner didn't actually prove the security of his scheme at the time he proposed it. Professor Aaronson asked about it on Stack Exchange a few years ago which prompted Molina, Vidick, and Watrous to write a paper that formally proved the scheme's security.

Lecture 8: Quantum Money and Quantum Key Distribution

8.1 Quantum Money Attacks

Last time we discussed how classical money is copyable and described a scheme for making money uncopyable through an application of the No-Cloning Theorem. Let's consider a counterfeiter who wants to take a copy of a legitimate bill B and submit it for verification. Say the counterfeiter decides to measure all qubits in the $\{|0\rangle, |1\rangle\}$ basis. They then make a new bill with the classical serial number copied and the quantum state given by the measurement results in the $\{|0\rangle, |1\rangle\}$ basis of the original state.

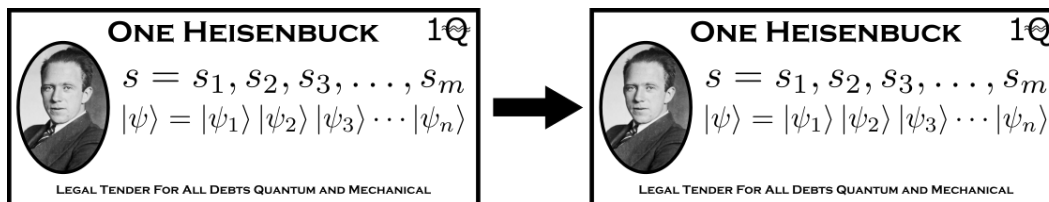


Figure 8.1: Possible result of counterfeiter trying to clone a bill by measuring each qubit in the standard basis.

When the bank goes to measure each qubit they'll find that the ones that should be in the $\{|0\rangle, |1\rangle\}$ basis are correct all of the time. But, the ones that should be in the $\{|+\rangle, |-\rangle\}$ basis are correct on both bills only $\frac{1}{4}$ of the time. Thus the probability that the counterfeiter succeeds (i.e., that both bills pass verification) is $(\frac{3}{8})^n$.

As we mentioned last time, it was shown in 2012 that any possible attack succeeds with probability at most $(\frac{3}{4})^n$.

8.1.1 Interactive Attacks

There's a clever attack on Wiesner's scheme based around the assumption that, after verification, the bank returns the bill *regardless of whether or not it passed verification*. We can start with a legitimate bill, then repeatedly go to the bank and ask them to verify it, manipulating each of the bill's qubits one at a time. For example, if we set the first qubit to $|0\rangle$ and the bill still passes verification each and every time then we've learned that the first qubit should be $|0\rangle$. Otherwise, we can successively try setting the first qubit to $|1\rangle$, $|+\rangle$ and $|-\rangle$ and see which choice makes the bank consistently happy. Then, once we know, we move on to toggling the second qubit and so on.

OK, but surely the bank wouldn't be so naïve as to return the bill even if it fails verification! We should assume instead that if verification fails (or fails often enough), then the bank alerts the police or something. *Can we come up with an attack that works even then?* A recent paper by Nagaj and Sattath points out that we can!

Recall the Elitzur-Vaidman Bomb discussed in Section 4.4. The general idea is that by making a succession of measurements, none of which reveals that much by itself, we can with a high probability of success learn whether a system is a certain state, without triggering the “bad event” that would happen if the system were actually measured to be in that state (such as a bomb going off). Nagaj and Sattath applied a similar idea to quantum money.

Elitzur-Vaidman Bomb Attack

Let $|\psi_i\rangle$ be the qubit of the banknote we want to learn. The protocol for the Elitzur-Vaidman bomb attack goes as follows:

- ▶ Initialize a qubit $|c\rangle$ to $|0\rangle$.
- ▶ Repeat $\frac{\pi}{2\epsilon}$ times:
 - Apply the rotation gate R_ϵ to $|c\rangle$.
 - Apply a CNOT gate to $|c\rangle|\psi_i\rangle$.
 - Send bill to the bank for verification

Suppose $|\psi_i\rangle = |0\rangle$. Then each time we apply CNOT, we get

$$CNOT(\cos(\epsilon)|0\rangle + \sin(\epsilon)|1\rangle)|0\rangle = \cos(\epsilon)|00\rangle + \sin(\epsilon)|11\rangle. \quad (8.1)$$

Following the measurement of the bill, most of the time $|c\rangle$ will snap back to $|0\rangle$. At each step the probability of getting caught (i.e. failing verification) is

$\sin^2(\epsilon) \approx \epsilon^2$. Thus the total probability of getting caught after the $\frac{\pi}{2\epsilon}$ iterations is upper-bounded by $\epsilon^2 \frac{\pi}{2\epsilon} = \mathcal{O}(\epsilon)$ by the union bound. A similar analysis can be done if $|\psi_i\rangle$ is $|1\rangle$ or $|-\rangle$; we're unlikely to get caught, and $|c\rangle$ keeps "snapping back" to $|0\rangle$. But if $|\psi_i\rangle = |+\rangle$ then something different happens; in that case the CNOT gate has no effect, so $|c\rangle$ gradually rotates from $|0\rangle$ to $|1\rangle$. So, when we measure at the end we can distinguish $|+\rangle$ from the other states because it's the only one that causes $|c\rangle$ to rotate to $|1\rangle$. By symmetry we can give analogous procedures to recognize the other three possible states for $|\psi_i\rangle$. So then we just iterate over all n qubits in the bill, learning them one by one just like in the previous interactive attack on Wiesner's scheme.

Can Wiesner's scheme be fixed to patch this vulnerability? Yes! The bank can just give the customer a new bill (of the same value) after each verification, instead of the bill that was verified.

There's an additional problem with Wiesner's scheme, as we've seen it. Namely, it requires the bank to hold a huge amount of information, one secret for every bill in circulation. However, a paper by Bennett, Brassard, Breidbart and Wiesner, from 1982, points out how to circumvent this by saying: let f be a pseudorandom function with a secret key k , so that for any serial number s , the bank can compute $f_k(s)$ for itself rather than needing to look it up. Of course the bank had better keep k itself secret—if it leaks then out the entire money system collapses! But assuming that k remains a secret, *why is this secure?*

We use a reduction argument. Suppose that the counterfeiter can copy money by some means. *What does that say about f_k ?* If f_k were truly random, then the counterfeiter wouldn't have succeeded, by the security of Wiesner's original scheme. So by checking whether the counterfeiter succeeds, we can distinguish f_k from a random function. So f_k wasn't very good at being pseudorandom! Note that with this change, we give up on information-theoretic security of the sort that we had with Wiesner's original scheme. Now we "only" have security assuming that it's computationally intractable to distinguish f_k from random. Moreover, a recent result by Prof. Aaronson shows that some computational assumption is *necessary* if we don't want the bank to have to store a giant database.

However, even after we make the improvements above, Wiesner's scheme still has a fundamental problem, which is that to verify a bill you need to take it back to the bank. If you have to go to the bank, then arguably you might as well have used a credit card or something instead! The point of cash is supposed to be that we don't need a bank to complete a transaction. This leads to the concept of **Public-Key Quantum Money**.

8.1.2 Public-Key Quantum Money

Public-Key Quantum Money refers to quantum money schemes in which anyone can verify a bill using a “public key,” but where a bill can only be produced or copied by using a “private key” known only to the bank. For formal definitions see (Aaronson 2009), (Aaronson, Christiano 2012). With this sort of scheme you’ll always need computational assumptions on the power of the counterfeiter in addition to the structure of quantum mechanics. *Why?* Because a counterfeiter with infinite computational power could always just try every possible quantum state (or an approximation thereof) on the appropriate number of qubits until it found one that made the public verification procedure accept.

8.2 Quantum Key Distribution

Now we’ll discuss something closely related to quantum money, but that doesn’t require storing quantum states for long times, and that for that reason is actually practical today (though so far there’s only a tiny market for it). Key distribution is a fundamental task in cryptography. It just means causing two agents, Alice and Bob, to share a secret key (without loss of generality, a uniformly random string) when they didn’t have one before. Once Alice and Bob share a long enough key, they can then exchange secret messages using the central technique in cryptography called the *One-Time Pad*, which works as follows:

- ▶ Given that Alice and Bob have a shared key $k \in \{0, 1\}^n$, Alice can take her secret message $m \in \{0, 1\}^n$ and encode it by the ciphertext $c = m \oplus k$, where \oplus denotes the bit-wise XOR.
- ▶ Bob, after receiving c , can decode the message using his copy of the secret key, using the fact that $c \oplus k = m \oplus k \oplus k = m$.

As its name implies, the One-Time Pad can only be used once securely with a given key, so it requires a large amount of shared key. In fact, in the classical world Claude Shannon proved that if they want to communicate securely, Alice and Bob either need a shared secret key that’s at least as long as all the messages that they want to send, or else they must make computational assumptions about the eavesdropper “Eve”. The great discovery of **Quantum Key Distribution** (QKD) was that quantum mechanics lets us get secure key distribution with no need for computational assumptions! We do, however, need communication channels capable of sending quantum states.

In cryptography, besides secrecy, an equally important goal is authentication. However, we're only going to deal with secrecy in this course.

The BB84 Protocol

In this section we'll describe the BB84 scheme, the first full quantum key distribution scheme. This scheme was proposed by Bennett and Brassard in 1984, though it was partly anticipated in Wiesner's paper (the same one that introduced quantum money!). It circumvents the issues we've seen in maintaining a qubit's coherence for a long time because it only requires coherence for the time it takes for communication between Alice and Bob.

There are companies and research groups that are already doing quantum key distribution through fiber optic cables over up to about 10 miles and through free-space nearly 90 miles. In addition, just a few years ago, in 2017, a team from China demonstrated QKD over distances of thousands of miles by sending photons to and from a satellite that was launched into space for that express purpose.

The basic idea is that you're trying to establish some shared secret knowledge and you want to know for certain that no eavesdroppers on the channel can uncover it. You've got a channel to transmit quantum information and a channel to transmit classical information. In both eavesdroppers may be able to listen in (no secrecy). But, in the classical channel we'll assume you at least have *authentication*; Bob knows that any messages really come from Alice and vice versa. The BB84 protocol proceeds as follows:

- ▶ Alice chooses uniformly at random a pair of strings $x, y \in \{0, 1\}^n$.
- ▶ Alice then generates an n -qubit state $|\psi\rangle$ where Alice uses the bits of y to determine which basis to encode her qubits in (0 for $\{|0\rangle, |1\rangle\}$ and 1 for $\{|+\rangle, |-\rangle\}$), and she uses the bits of x to determine the element of that basis ($0 \rightarrow |0\rangle / |+\rangle$ and $1 \rightarrow |1\rangle / |-\rangle$).
- ▶ Alice sends the quantum state $|\psi\rangle$ to Bob.
- ▶ Bob picks a string y' uniformly at random from $\{0, 1\}^n$.
- ▶ Bob uses the bits of y' to determine the basis in which to measure each of the qubits sent from Alice. He then records the results of the measurements in the string x' ($|0\rangle / |+\rangle \rightarrow 0$ and $|1\rangle / |-\rangle \rightarrow 1$).

- ▶ Now Alice and Bob share which bases they picked to encode and measure the state $|\psi\rangle$ (the strings y and y'). They discard any bits of x and x' for which they didn't pick the same basis (which will be about half the bits). What remains of x and x' is now their shared secret key.

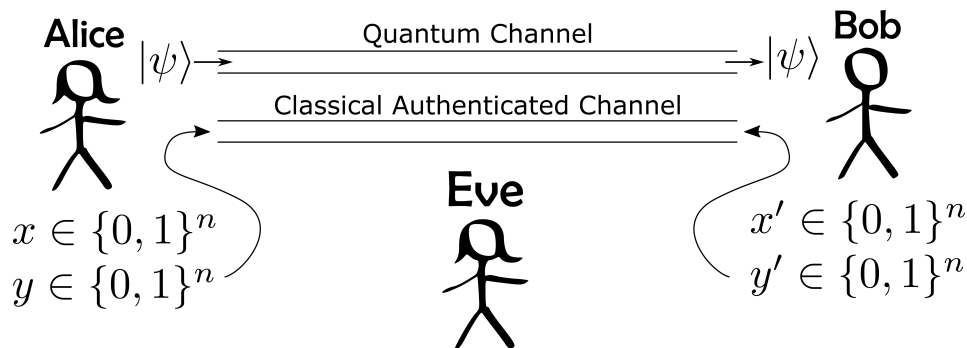


Figure 8.2: Sketch of the BB84 protocol.

In Figure 8.2 we roughly sketch the protocol visually.

At this point we consider an eavesdropper Eve who was watching the qubits as they were sent over. The whole magic of using qubits is that if Eve tries to measure the qubits then she inherently changes what Bob receives! Sure, if she measures a $|0\rangle$ or $|1\rangle$ and the qubit was prepared in the $\{|0\rangle, |1\rangle\}$ basis then the qubit doesn't change. But, what if she's unlucky and measures a qubit in a basis it wasn't prepared in? Eventually she almost certainly *will* be unlucky and have this happen.

In more detail, suppose Alice sent $|+\rangle$ and Eve measures in the wrong basis and sees $|0\rangle$, which gets passed along to Bob. Then, even if Bob measures in the $\{|+\rangle, |-\rangle\}$ basis (i.e., the “right” basis), he has a 50% chance of measuring $|+\rangle$ and a 50% chance of measuring $|-\rangle$. In the latter case Alice and Bob will be able to see that the channel was tampered with. So, Alice and Bob can verify that no one listened in to their qubit transmission by making sure that the portion of their qubits that *should* match, do match. Of course, after Alice and Bob discuss those qubits over the channel they aren't going to be secret anymore! But they've still got all the others. If a large enough fraction of the qubits didn't match then Alice and Bob deduce that Eve eavesdropped. So then they can just keep trying again and again until they can get a batch where no one listened in. At worst, Eve can prevent Alice and Bob from ever communicating by listening in constantly. But we can prevent a situation where Alice and Bob think their shared key is secure even though it isn't. Again, once Alice and Bob share a secret key, they can then use some

classical encryption scheme, like the One-Time Pad, or a scheme that depends on computational assumptions (if they want to make their shared secret key last longer).

Lecture 9: Superdense Coding

9.1 Superdense Coding

Superdense Coding is the first protocol we'll see that requires entanglement. Basic information theory (Shannon) tells us that “by sending n bits, you can't communicate more than n bits of information.” Now, by contrast, we'll see how Alice can send Bob two classical bits by sending him only one qubit, though there is a catch: Alice and Bob must share some entanglement ahead of time. In the scenario with no prior entanglement, Alice can't send more than one bit per qubit—a fundamental result known as **Holevo's Theorem**. We're not going to prove Holevo's theorem here, but the intuition is pretty simple; if Alice sends $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ to Bob he can only measure it once in some basis and then the rest of the information in $|\psi\rangle$ is lost.

Instead, let's suppose that Alice and Bob share a Bell pair in advance $|\psi\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}}$. We claim that Alice can manipulate her half, then send her half to Bob and finally Bob can measure both qubits and get two bits of information from Alice. The key is to realize that Alice can get three different states, all of them orthogonal to the original Bell pair and to each other, by applying the following gates to her qubit

$$\begin{aligned}(X \otimes I) \left(\frac{|00\rangle + |11\rangle}{\sqrt{2}} \right) &= \left(\frac{|01\rangle + |10\rangle}{\sqrt{2}} \right) \\(Z \otimes I) \left(\frac{|00\rangle + |11\rangle}{\sqrt{2}} \right) &= \left(\frac{|00\rangle - |11\rangle}{\sqrt{2}} \right) \\(Z \otimes I)(X \otimes I) \left(\frac{|00\rangle + |11\rangle}{\sqrt{2}} \right) &= \left(\frac{|01\rangle - |10\rangle}{\sqrt{2}} \right)\end{aligned} \tag{9.1}$$

These three states, together with $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$, form an orthonormal basis. So, suppose Alice wants to transmit two bits x , and y :

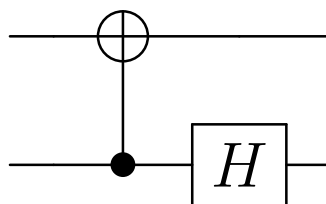
- If $x = 1$, she applies the X gate.

- ▶ If $y = 1$, she applies a Z gate.
- ▶ Then she sends her qubit to Bob.

For Bob to decode this transformation, he'll want to use the transformation

$$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & -1 & 0 \end{bmatrix}, \quad (9.2)$$

which corresponds to the circuit



So, Alice transforms the Bell pair into one of the four orthogonal states above, then Bob decodes that two-qubit state into one of the four possible combinations of $|0\rangle$ and $|1\rangle$, corresponding to the original bits x and y . For example, if Bob receives $\frac{|01\rangle - |10\rangle}{\sqrt{2}}$, then applying CNOT gets him $|1\rangle \otimes |-\rangle$ and then Hadamard gives him $|1\rangle \otimes |1\rangle$. If Bob receives $\frac{|00\rangle - |11\rangle}{\sqrt{2}}$, then applying CNOT gets him $|0\rangle \otimes |+\rangle$ and the Hadamard then gives him $|0\rangle \otimes |1\rangle$.

Naturally, we could ask: if Alice and Bob had even more pre-shared entanglement, *could Alice send an arbitrarily large amount of information by transmitting only one qubit?* There's a theorem that says *no*. It turns out that given a qubit and any number of pre-shared entangled qubits (ebits), you can send two bits of classical information, but no more. That is, we can write the inequality

$$1 \text{ qubit} + 1 \text{ ebit} \geq 2 \text{ bits}, \quad (9.3)$$

but, we can't write $1 \text{ qubit} + \text{ebits} \geq k \text{ bits}$ for any $k > 2$. As far as quantum speed-ups go, a factor of two isn't particularly impressive, but it is pretty cool that it challenges the most basic rules of information theory established by Shannon himself.

Lecture 10: Teleportation, Entanglement Swapping, GHZ State and The Monogamy of Entanglement

10.1 Quantum Teleportation

Quantum Teleportation is a foundational discovery from 1991 that came as a great surprise. Science journalists still love it given its irresistible name. In this lecture we'll see what it can and can't do. Firstly, *what does teleportation mean?* You might think it implies sending qubits instantaneously over vast distances, but that can't be done, as it violates the causal structure of the universe dictated by the laws of special relativity. So we're only going to send qubits at most at the speed of light, no faster. Of course, there are other ways to move qubits at the speed of light or slower, like just picking them up and moving them, or putting them on a bus! (It doesn't sound as sexy that way.) OK, but what if you only had a phone line, or a standard Internet connection? That would let you send classical bits, but not qubits. With teleportation, though, we'll achieve something surprising. We'll show that it's possible for Alice and Bob to use pre-shared entanglement plus classical communication to perfectly transmit a qubit.

The inequality here is almost the converse of the one for superdense coding:

$$1 \text{ ebit} + 2 \text{ bits} \geq 1 \text{ qubit} \tag{10.1}$$

Which is to say, you need one pair of entangled qubits plus two classical bits in order to transmit one qubit. This can also be shown to be optimal. So, let's say Alice wants to get a qubit over to Bob, without using a quantum communication channel, but with a classical channel together with pre-shared

entanglement. *How should Alice go about this?* Once the question is posed, you can play around with different combinations of operations and you'd eventually discover that what works is this:

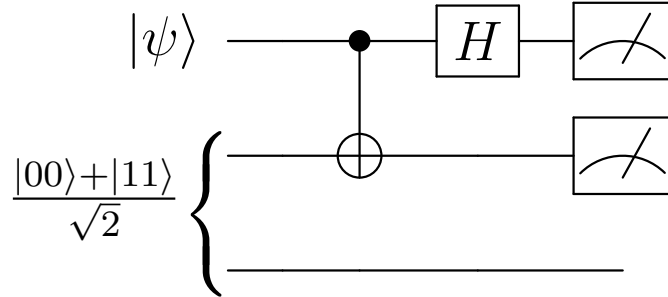


Figure 10.1: Quantum circuit for performing quantum teleportation protocol.

where $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ is the state Alice wishes to send. The top two qubits in the circuit above are Alice's. *At the end, will Alice also have $|\psi\rangle$?* No. A logical consequence of the No-Cloning Theorem is that there can only be one copy of the qubit. *Could we hope for a similar protocol without sending classical information?* No, because of the No-Communication Theorem.

Now let's analyze the behavior of the circuit in Figure 10.1 in more detail. The qubit Alice wants to transmit is $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. The combined state of her qubit, along with the entangled Bell Pair she shares with Bob is

$$(\alpha|0\rangle + \beta|1\rangle) \otimes \frac{|00\rangle + |11\rangle}{\sqrt{2}} = \frac{\alpha|000\rangle + \alpha|011\rangle + \beta|100\rangle + \beta|111\rangle}{\sqrt{2}}. \quad (10.2)$$

Following the CNOT the state of the system is

$$\frac{\alpha|000\rangle + \alpha|011\rangle + \beta|110\rangle + \beta|101\rangle}{\sqrt{2}}. \quad (10.3)$$

Next, Alice applies a Hadamard to her first qubit which results in the state

$$\begin{aligned} & \frac{1}{\sqrt{2}} (\alpha|+00\rangle + \alpha|+11\rangle + \beta|-10\rangle + \beta|-01\rangle) \\ &= \frac{1}{2} (\alpha|000\rangle + \alpha|100\rangle + \alpha|011\rangle + \alpha|111\rangle + \beta|010\rangle - \beta|110\rangle + \beta|001\rangle - \beta|101\rangle) \end{aligned} \quad (10.4)$$

Finally, Alice measures both of her qubits in the $\{|0\rangle, |1\rangle\}$ basis. This leads to four possible outcomes for the state of Bob's qubit conditioned on her

measurement results, as shown in Table 10.1. We’re deducing information about by Bob’s state by using the partial measurement rule. E.g., if Alice sees $|00\rangle$, then we narrow down the state of the entire system to the possibilities that fit, namely $|000\rangle$ and $|001\rangle$.

If Alice Sees:	00	01	10	11
Then Bob’s qubit is:	$\alpha 0\rangle + \beta 1\rangle$	$\alpha 1\rangle + \beta 0\rangle$	$\alpha 0\rangle - \beta 1\rangle$	$\alpha 1\rangle - \beta 0\rangle$

Table 10.1: Summary of Bob’s output state conditioned on Alice’s measurement results.

What is Bob’s state, if he knows that Alice measured, but doesn’t know the measurement outcome? It’s an equal mixture of all four possibilities, which is just the Maximally Mixed State. This makes sense given the No-Communication Theorem! Until Alice sends information over, Bob’s qubit can’t possibly depend on $|\psi\rangle$.

Next, Alice tells Bob her measurement results via a classical channel and Bob uses the information to “correct” his qubit to $|\psi\rangle$. If the first bit sent by Alice is 1 then Bob applies Z , and if the second bit sent by Alice is 1 then Bob applies X . These transformations will bring Bob’s qubit to the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. That means they’ve successfully transmitted a qubit without a quantum channel!

Note this protocol never assumed that Alice knew what $|\psi\rangle$ was.

For the protocol to work, Alice had to measure her *syndrome* bits and communicate the result to Bob. These measurements were destructive (since we can’t ensure that they’ll be made in a basis orthonormal to $|\psi\rangle$), and thus Alice doesn’t have $|\psi\rangle$ at the end. Alice and Bob also “use up” their Bell pair in the process of teleporting $|\psi\rangle$.

Something to think about: Where is $|\psi\rangle$ after Alice’s measurement, but before Bob does his operations?

FAQ***How do people come up with this stuff?***

Well it's worth pointing out that quantum mechanics was discovered in 1926 and that quantum teleportation was only discovered in the 90's. These sorts of protocols can be hard to find. Sometimes someone tries to prove that something is impossible, and in doing so eventually figures out a way to get it done. . .

Aren't we fundamentally sending infinitely more information than two classical bits if we've sent over enough information to perfectly describe an arbitrary qubit, since the qubit's amplitudes could be arbitrary complex numbers?

In some sense, but at the end of the day, Bob only really obtains the information that he can measure, which is significantly less. Amplitudes may "exist" physically, but they're different from other physical quantities like length, in that they seem to act a lot more like probabilities. Like, there's a state of a single qubit $\alpha|0\rangle + \beta|1\rangle$ such that the binary encoding of β corresponds to the complete works of Shakespeare—the rules of quantum mechanics don't put a limit on the amount of information that it takes to specify an amplitude. With that said, we could also encode the complete works of Shakespeare into the probability that a classical coin lands heads! In both cases, the works of Shakespeare wouldn't actually be retrievable by measuring the system, assuming we didn't have an immense number of copies of it.

10.1.1 Multi-Qubit Teleportation and Entanglement Swapping

Can we go further? What would it take to teleport an arbitrary quantum state, say of n qubits? To answer this question, let's notice that nothing said that a qubit that's teleported has to be unentangled with the rest of the world. You could run the protocol and have $|\psi\rangle$ be half of another Bell pair. That would entangle the fourth qubit to Bob's qubit (you can check this via calculation). This operation is depicted in Figure 10.2. This is not a particularly interesting operation since it lands you where you started with one qubit of entanglement between Alice and Bob, but it does have an interesting implication. It suggests that it should be possible to teleport an arbitrary n -qubit entangled state, by simply teleporting the qubits one at a time, thus using n ebits of preshared entanglement and $2n$ bits of classical communication. Indeed, it's not hard to check that this works.

One further consequence of this is that two qubits don't need to interact

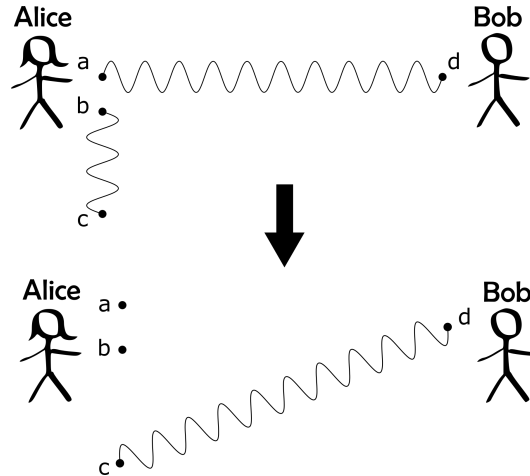
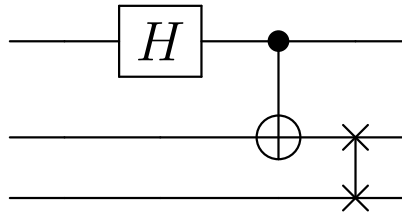


Figure 10.2: Diagrammatic depiction of teleportation in which Alice uses one ebit of pre-shared entanglement with Bob to teleport over one half of some arbitrary entangled state that she has control of.

directly to become entangled. In some sense, we already knew that. Consider, for example, the following circuit:



The final gate in this circuit is a SWAP gate between the last two qubits. Note that the first and third end up entangled even though there's never “direct” contact between them. The second qubit serves as an intermediary.

What does it take for Alice and Bob to get entangled? The obvious way is for Alice to create a Bell pair and then send one of the qubits to Bob. In most real-world experiments the entangled qubits are created somewhere between Alice and Bob and then one qubit is sent to each.

Anyway, teleportation leads to a more surprising protocol than this, called **Entanglement Swapping**. Imagine we have four parties: Alice, Bob, Charlie and Diane. Alice and Bob share a Bell pair, as do Alice and Charlie and Charlie and Diane, as depicted in Figure 10.3. Now suppose that Alice teleports her half of the Bell pair that she shares with Charlie to Bob, and that Charlie

teleports his half of the Bell pair that he shares with Alice to Diane. The result of this series of teleportations is that Bob and Diane now both have one half of a Bell pair—even though the two qubits Bob and Diane possess were never in causal contact with one another!

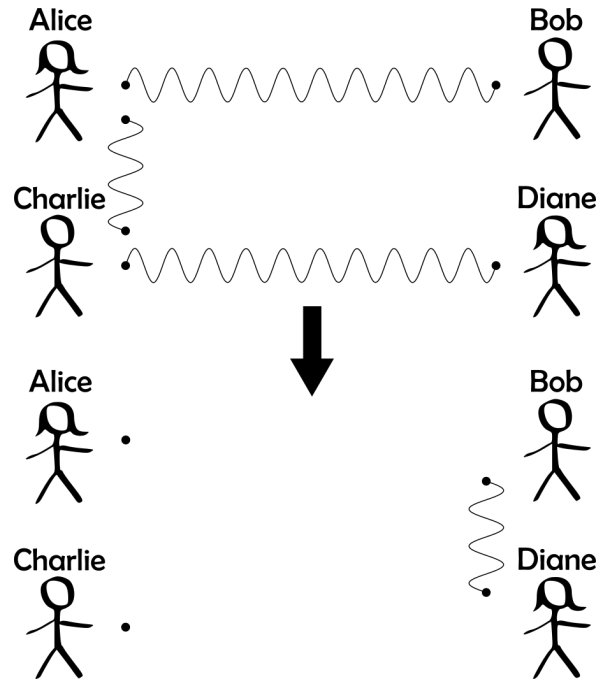


Figure 10.3: Entanglement swapping procedure depicted. Initially Alice and Bob share a Bell pair, as do Charlie and Diane and Alice and Charlie. Following a pair of teleportations from Alice to Bob and from Charlie to Diane we find that Bob and Diane now share a Bell pair (even though they never interacted directly).

This process has been used in real experiments, such as the recent “loophole-free Bell tests,” which we’ll learn about later in the course. Quantum teleportation itself has been demonstrated experimentally many times.

10.1.2 The GHZ State and Monogamy of Entanglement

We’ve seen the Bell pair, and what it’s good for. There’s a 3-qubit analogue of it called the GHZ state, $\frac{|000\rangle + |111\rangle}{\sqrt{2}}$. We’ll see the GHZ state again later in the

course, but for now we'll use it to illustrate an interesting conceptual point. Let's say that Alice, Bob, and Charlie hold random bits, which are either all 0 or all 1 (so, they're classically correlated). If all three of them get together, they can see that their bits are correlated and the same is true *even if only two of them are together*.

Now suppose instead that the three players share a GHZ state. With all three of them together they can see that the state is entangled, but what if Charlie is gone? *Can Alice and Bob see that they're entangled with each other?* No. To see this observe that by the No-Communication Theorem, Charlie could've measured without Alice and Bob knowing. But, if he did, then Alice and Bob would clearly have classical correlation only: either both $|0\rangle$'s (if Charlie got the measurement outcome $|0\rangle$) or both $|1\rangle$ (if Charlie got $|1\rangle$). From this it follows that Alice and Bob have only classical correlation *regardless of whether Charlie measured or not*.

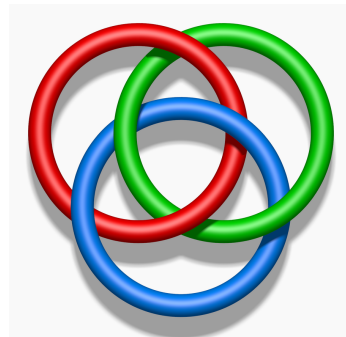
A different way to see this is to look at the reduced density matrix of the state shared by Alice and Bob,

$$\rho^{AB} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} \end{bmatrix}.$$

Notice that this is different than the density matrix of a Bell pair,

$$\rho_{\text{Bell}} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{bmatrix}.$$

This is one illustration of a general principle called the **Monogamy of Entanglement**. Simply put, if Alice has a qubit that is maximally entangled with Bob's qubit, then that qubit can't also be maximally entangled with Charlie's qubit. With the GHZ state, you can only see the entanglement if you have all three qubits together. This is sometimes analogized to the Borromean Rings, an arrangement of three rings with the property that all three are linked together, but removing any one ring unlinks the other two.



There are other 3-qubit states that behave differently than the GHZ state. In the W state, $\frac{|100\rangle + |010\rangle + |001\rangle}{\sqrt{3}}$, there's *some* entanglement between Alice and

74 LECTURE 10. TELEPORTATION, ENTANGLEMENT SWAPPING...

Bob, and there's *some* entanglement between Alice and Charlie, but neither pair is *maximally* entangled.

In the next lecture, we'll make more rigorous precisely what we mean by saying Alice and Bob share "some" entanglement when we talk about how to quantify entanglement.

Lecture 11: Quantifying Entanglement

How do you quantify how much entanglement there is between two quantum systems? It's worth noting that we get to decide what we think a measure of entanglement ought to mean. We've seen how it can be useful to think of Bell pairs as a resource, so we can phrase the question as "how many 'Bell pairs of entanglement' does a given state correspond to?"

A priori, there could be different incomparable kinds of entanglement that are good for different things. That's actually the case for entangled mixed states, or entangled pure states shared by three or more parties. But, for the special case of an entangled pure state shared by two parties, it turns out that there's a single measure of entanglement which counts the number of Bell pairs needed to form the state, or equivalently the number of Bell pairs that can be extracted from it.

So, given a bipartite state

$$\sum_{ij} \alpha_{ij} |i\rangle_A |j\rangle_B, \quad (11.1)$$

how do we calculate many Bell pairs it's worth? Our first observation is that given any bipartite pure state you can always find a change of basis on Alice's side and another change of basis on Bob's side that puts the state into the simpler form

$$\sum_i \lambda_i |v_i\rangle |w_i\rangle, \quad (11.2)$$

where the set of states $\{|v_i\rangle\}$ form an orthonormal basis and likewise for the set

of states $\{|w_i\rangle\}$ —though the sets of states $\{|v_i\rangle\}$ and $\{|w_i\rangle\}$ are not necessarily orthonormal with respect to each other. We call the form of the state in Equation 11.2 the **Schmidt Decomposition** or **Schmidt Form**.

11.1 Schmidt Decomposition

Suppose we have a quantum state written in the form $\sum_{ij} \alpha_{ij} |i\rangle_A |j\rangle_B$. Then we can rewrite the coefficients α_{ij} in matrix form, given by

$$A = \begin{bmatrix} \alpha_{0,0} & \alpha_{0,1} & \cdots & \alpha_{0,n-1} \\ \alpha_{1,0} & \alpha_{1,1} & & \\ \vdots & & \ddots & \\ \alpha_{n-1,0} & & & \alpha_{n-1,n-1} \end{bmatrix}. \quad (11.3)$$

We can then use the **Singular Value Decomposition (SVD)** to rewrite our matrix as $A = U\Lambda V^\dagger$, where U and V are $n \times n$ unitary matrices, and Λ is an $n \times n$ diagonal matrix whose entries we call the *singular values* of A . Rearranging the factorization we find that $\Lambda = U^\dagger A V$, and so the changes of basis which Alice and Bob would need to perform in order to get their state into Schmidt form are given by U^\dagger and V respectively.

Note that while we won't prove it here there are efficient algorithms for calculating the SVD of a matrix and these are available out-of-the-box in any linear algebra software.

Measuring in the $\{|v_i\rangle |w_i\rangle\}$ basis would yield the probability distribution $[|\lambda_0|^2, \dots, |\lambda_{n-1}|^2]^\top$.

11.2 Von Neumann Entropy

Recall that for a classical probability distribution $P = [p_0, \dots, p_{n-1}]^\top$, its Shannon entropy is

$$H(P) = \sum_{i=0}^{n-1} p_i \log_2 \left(\frac{1}{p_i} \right). \quad (11.4)$$

There's a generalization of this measure that works for quantum states

(both pure and mixed), called the **von Neumann Entropy**. The von Neumann entropy of a mixed state ρ is

$$S(\rho) = \sum_{i=0}^{n-1} \gamma_i \log_2 \left(\frac{1}{\gamma_i} \right) \quad (11.5)$$

where $\{\gamma_i\}$ are the eigenvalues of ρ . If you diagonalize a density matrix, the diagonal represents a probability distribution over n possible outcomes (corresponding to the eigenstates of ρ), and taking the Shannon entropy of that distribution gives you the von Neumann entropy of your quantum state. Note that physicists often use an alternate convention for the von Neumann entropy using the natural logarithm rather than base-2 (we'll always use base-2 in this course).

Another way we can think about the von Neumann entropy is as follows. Say you looked at all the possible probability distributions that could arise by measuring the mixed state ρ in all possible orthogonal bases. Then the von Neumann entropy of ρ is the minimum of the Shannon entropies of all those distributions,

$$S(\rho) = \min_U H(\text{diag}(U\rho U^\dagger)), \quad (11.6)$$

where $\text{diag}(A)$ is the length- n vector obtained from the diagonal of the $n \times n$ matrix A . This alternative definition makes it immediately clear that the von Neumann entropy of any pure state $|\psi\rangle$ is 0, because there's always some measurement basis (namely, a basis containing $|\psi\rangle$) that returns a definite outcome.

For example, you could choose to measure the state $|+\rangle$ in the $\{|0\rangle, |1\rangle\}$ basis and you'll have complete uncertainty and a Shannon entropy of 1. But, if you measure $|+\rangle$ in the $\{|+\rangle, |-\rangle\}$ basis you'll have a Shannon entropy of 0 because you'll always get the outcome $|+\rangle$. As such, the von Neumann entropy of $|+\rangle$ is 0. By contrast, the von Neumann entropy of the maximally mixed state, $\frac{I}{2}$, is 1; similarly, the von Neumann Entropy of the n -qubit maximally mixed state is n .

11.2.1 Entanglement Entropy

We can now begin to answer the question posed at the start of this lecture: *how do you quantify how much entanglement there is between two quantum*

systems? Suppose Alice and Bob share a bipartite pure state

$$|\psi\rangle = \sum_{ij} \alpha_{ij} |i\rangle_A |j\rangle_B. \quad (11.7)$$

To quantify the entanglement of this state we'll use a measure called the **Entanglement Entropy**. The entanglement entropy of a (pure) bipartite state is given by

$$E(|\psi\rangle) = S(\rho^A) = S(\rho^B) = H([\lambda_0|^2, \dots, |\lambda_{n-1}|^2]^\top) \quad (11.8)$$

where ρ^A and ρ^B are the reduced density matrices corresponding to Alice and Bob's subsystems respectively and where $[\lambda_0|^2, \dots, |\lambda_{n-1}|^2]^\top$ is the probability distribution corresponding to the Schmidt form of Alice and Bob's shared state. This definition has the desirable properties that the entanglement entropy of any product state $|\psi\rangle \otimes |\phi\rangle$ is 0 and the entanglement entropy of a Bell pair $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$ is 1.

You can think of the entanglement entropy of a state as either the number of Bell pairs it would take to create it or as the number of Bell pairs that you can extract from it. It's not immediately obvious that these numbers are the same, but for pure states they are. For mixed states they need not be!

As an example, suppose Alice and Bob share the state

$$|\psi\rangle = \frac{3}{5} |0\rangle |+\rangle + \frac{4}{5} |1\rangle |-\rangle.$$

Then we can calculate the entanglement entropy using any of the three equivalent definitions in Equation 11.8. In this case we already have the state written in Schmidt form, so we can directly use the last definition in Equation 11.8:

$$E(|\psi\rangle) = \left(\frac{3}{5}\right)^2 \log_2 \left(\left(\frac{5}{3}\right)^2\right) + \left(\frac{4}{5}\right)^2 \log_2 \left(\left(\frac{5}{4}\right)^2\right) \approx .942.$$

This means that if Alice and Bob shared 1000 copies of $|\psi\rangle$, they'd be able to teleport about 942 qubits. We can also confirm that the other two definitions for the entanglement entropy, in terms of the von Neumann entropies of Alice and Bob's reduced density matrices, do in fact give the same value. Suppose Bob measures his state in the $\{|+\rangle, |-\rangle\}$ basis. Bob sees $|+\rangle$ with probability $9/25$, in which case Alice's qubit is $|0\rangle$, and he sees $|-\rangle$ with probability $16/25$, in which case Alice's qubit is $|1\rangle$. As such, the reduced density matrix for Alice's qubit is

$$\rho^A = \frac{9}{25} |0\rangle \langle 0| + \frac{16}{25} |1\rangle \langle 1|.$$

Likewise, suppose Alice measures her qubit in the $\{|0\rangle, |1\rangle\}$ basis. Alice will see $|0\rangle$ with probability $9/25$, in which case Bob's qubit is $|+\rangle$, and she'll see $|1\rangle$ with probability $16/25$, in which case Bob's qubit is $|-\rangle$. As such, the reduced density matrix for Bob's qubit is

$$\rho^B = \frac{9}{25} |+\rangle \langle +| + \frac{16}{25} |-\rangle \langle -|.$$

Both of these reduced density matrices are already diagonalized and so we can immediately see that they share the same eigenvalues—though they have different eigenvectors. Moreover, the values of the eigenvalues correspond exactly to the values of the squared coefficients of the Schmidt form of the state. As such, we'll find precisely the same value for the entanglement entropy regardless of whether we calculate it by finding the Schmidt form of a given state or by finding one of the reduced density matrices. In practice, this means you should feel free to use whichever method is most convenient.

11.3 Mixed State Entanglement

For bipartite mixed states, there are two values to consider. The **Entanglement of Formation**, $E_F(\rho)$, is the number of ebits needed per copy of ρ created in the asymptotic limit of a large number of copies of ρ . We assume Alice and Bob are allowed unlimited local operations and classical communication (called “LOCC” in the lingo) for free in any protocols they use. The **Distillable Entanglement**, $E_D(\rho)$, is the number of ebits that Alice and Bob can extract per copy of ρ in the asymptotic limit where they have many copies of ρ , again assuming LOCC is free.

Clearly $E_F \geq E_D$, since if you could ever get out more entanglement than you put in it would give you a way to increase entanglement arbitrarily using LOCC, which is easily seen to be impossible. *But, what about the other direction?* It turns out that there exist bipartite pure states for which $E_F \gg E_D$, which is to say that those states take a lot of entanglement to make, but then you can only extract a small fraction of the entanglement that you put in. There are even states (called *bound entangled states*) for which $E_F > 0$, but $E_D = 0$. A detailed treatment of such states is beyond the scope of this course.

We call a bipartite mixed state ρ *separable* if there's *any* way to write it as a mixture of product states,

$$\rho = \sum_i p_i |v_i\rangle \langle v_i| \otimes |w_i\rangle \langle w_i|. \quad (11.9)$$

A mixed state is called entangled if and only if it's not separable. This condition is subtle, as it sometimes happens that a density matrix *looks* entangled, but it turns out there's some non-obvious decomposition which shows it's actually separable. It's important to note that the converse of the separability criterion in Equation 11.9 is not true. That is, being able to decompose a mixed state into a convex combination of entangled states does *not* imply that the mixed state is entangled. A simple counterexample is the two-qubit maximally mixed state

$$\frac{I}{4} = \frac{|00\rangle \langle 00| + |01\rangle \langle 01| + |10\rangle \langle 10| + |11\rangle \langle 11|}{4}. \quad (11.10)$$

This state clearly satisfies the criterion in Equation 11.9, and so this state is separable. However, we can also write the maximally mixed state as

$$\frac{I}{4} = \frac{1}{4} \sum_{i=0}^3 |\phi_i\rangle \langle \phi_i|,$$

where the $|\phi_i\rangle$'s are each one of the 4 Bell states

$$\begin{aligned} |\phi_0\rangle &= \frac{|00\rangle + |11\rangle}{\sqrt{2}} & |\phi_1\rangle &= \frac{|00\rangle - |11\rangle}{\sqrt{2}} \\ |\phi_2\rangle &= \frac{|01\rangle + |10\rangle}{\sqrt{2}} & |\phi_3\rangle &= \frac{|01\rangle - |10\rangle}{\sqrt{2}} \end{aligned}$$

In a sense we got some initial intuition about the difficulty of this problem in Lectures 6 and 7, where we saw that even for the case of a one-qubit mixed state there are an infinite number of decompositions as convex combinations of pure states. Indeed, in 2003 Leonid Gurvits proved a pretty crazy fact: if you're given as input a density matrix ρ for a bipartite state, then deciding whether ρ represents a separable or entangled state is an **NP**-hard problem! As a result, unless $\mathbf{P} = \mathbf{NP}$, there can be no “nice characterization” for telling apart entangled and separable bipartite mixed states—in contrast to the situation with bipartite pure states.

This helps to explain why there are endless paper writing

opportunities in trying to classify different types of entanglement...

Lecture 12: Interpretations of Quantum Mechanics

At this point in the course, we're finally in a position to step back and ask, *what is quantum mechanics telling us about reality?* It should be no surprise that there isn't a consensus on this question (to put it mildly)! But, regardless of your own views, it's important to know something about the various positions people have defended over the years, as the development of these positions has sometimes gone hand in hand with breakthroughs in quantum mechanics. We'll see an example of this later with the Bell inequality, and arguably quantum computing itself is another example. Most discussions about the implications of quantum mechanics for our understanding of reality center around the so-called **Measurement Problem**. In most physics texts (and in this class for that matter) measurement is introduced as just a primitive operation whose implications we don't try to understand more deeply. However, there's a fundamental weirdness about measurement in QM, which stems from the fact that the theory seems to demand both:

- ▶ Unitary evolution in which $|\psi\rangle \rightarrow U|\psi\rangle$.
- ▶ Measurements in which a state collapses to some outcome $|i\rangle$ with probability $|\langle\psi|i\rangle|^2$.

In other words, quantum mechanics seems to work in a way that's deterministic, reversible, and continuous most of the time, *except* during measurement which is the only time we see it work in a way that's probabilistic, irreversible, and discontinuous. So we can phrase the question as: *How does the universe know when to apply unitary evolution and when to apply measurement?* People have argued about the correct interpretation of quantum mechanics for almost 100 years. The discussion is sometimes compared to the discussion about the nature of consciousness (which has gone on for millennia) in that they both tend to devolve into people talking in circles around each other. But, it's nonetheless worthwhile to understand the main schools of thought.

12.1 The Copenhagen Interpretation

The **Copenhagen Interpretation** was the preferred interpretation of most of the founders of quantum mechanics. It's closely associated with Niels Bohr and his institute in Copenhagen (hence the name) and with Werner Heisenberg. Note that the different founders said different and sometimes contradictory things, sometimes in very abstruse language, so it's notoriously hard to pin down what the "Copenhagen Interpretation" actually is! Essentially though, the Copenhagen viewpoint is that there are two different worlds (or parts of reality): the quantum world and the classical world. We live in the classical world where objects have definite locations and states and where objects can be measured without significantly altering them. But, in doing experiments we've discovered that there also exists the quantum world "beneath" ours which obeys very different rules. Measurement in this view is the operation that bridges the two worlds. It lets us "peek under the hood" into the quantum world and see what's going on.

Bohr wrote long tracts in which he argues that even making statements about the outcomes of quantum measurements presupposes that there must be a classical world in which those statements can be sensibly made. So, there's some "boundary" or "cut" between the quantum and classical worlds. The exact location of this boundary might be fuzzy and might vary depending on what sort of question we're asking. In any case, we should never make the error of insisting that our commonsense classical concepts remain valid on the quantum side of the boundary. Believers in the Copenhagen interpretation tend to say things like "if this doesn't make sense to you, then you're just stuck in the old classical way of thinking, and you need to change; the problem is not with quantum mechanics, it's with you."

12.1.1 "Shut Up and Calculate"

Our next option is closely related to the Copenhagen interpretation. This is probably one of the preferred "interpretations" for many physicists, chemists and others who work with quantum mechanics. It says that at the end of the day quantum mechanics works and it correctly predicts the results of experiments. This is all we can reasonably ask of a scientific theory, or at least all that it's fruitful to ask. Prof. Aaronson likes to say that the Copenhagen interpretation is basically just shut up and calculate (SUAC) without the SU part! Copenhagen starts from the idea that "it's pointless to philosophize about what this means," but then elevates that to a philosophy which of course is a little ironic.

While the SUAC view has some obvious practical advantages, it seems clear that it can't satisfy people's curiosity forever. This is not only because science has always aspired to understand what the world is like, with experiments and predictions a means to that end. A second reason is that as experimenters become able to create ever larger and more complicated quantum superpositions—in effect “breaching” the Copenhagen boundary between the quantum and classical worlds—it becomes less and less viable to “quarantine” quantum mechanics as simply a weird mathematical formalism that happens to work for predicting the behavior of electrons and photons. The more QM impinges on the world of our everyday experience, the more it seems necessary to come to terms with whatever it says about that world.

12.2 Schrödinger's Cat and Wigner's Friend

There were physicists in the 20's and 30's who never accepted the Copenhagen interpretation, the most famous of whom were Einstein and Schrödinger. Between them they came up with various thought experiments designed to try and show just how untenable it is to have a rigid boundary between the quantum and classical worlds. By far the most famous example is **Schrödinger's Cat**, which first appears with Einstein remarking in a letter that if you think of a pile of gunpowder as being inherently unstable you could model it as a quantum state which looks like

$$\frac{|\text{💣}\rangle + |\text{💣💣}\rangle}{\sqrt{2}}$$

Then, Schrödinger adds some flair by asking *what happens if we create a quantum state that corresponds to a superposition of states, in one of which a cat is alive and the other the cat is dead?* (Or perhaps a superposition of happy and sad, $\frac{1}{\sqrt{2}}(|\text{😺}\rangle + |\text{😿}\rangle)$, if you prefer a less grisly thought experiment). He isolates the state of the cat from the external environment by putting it in a box. The point of Einstein and Schrödinger's thought experiment is that the formal rules of quantum mechanics apply whenever you have distinguishable states, regardless of their size. In particular, they say that in principle you can create arbitrary linear combinations of such states. By the time we're talking about something as big as a cat, it seems patently obvious that we should have to say something about the nature of what's going on before measurement. Otherwise we'd devolve into extreme solipsism—saying, for example, that the cat only exists once we've opened the box to observe it.

Wigner’s Friend is a similar thought experiment proposed by the theoretical physicist Eugene Wigner in 1961. Suppose that Wigner could be put into an equal superposition of thinking one thought and thinking another one (say wanting either pancakes or eggs for breakfast), which we model as

$$\frac{|\text{Wigner}_0\rangle + |\text{Wigner}_1\rangle}{\sqrt{2}} = \frac{|\text{pancakes}\rangle + |\text{eggs}\rangle}{\sqrt{2}}.$$

Now, consider the joint state of Wigner and a friend who is capable of measuring the state of Wigner’s mind, but hasn’t done so yet.

$$\begin{aligned} |\text{Wigner’s Friend}\rangle \otimes \frac{|\text{Wigner}_0\rangle + |\text{Wigner}_1\rangle}{\sqrt{2}} = \\ \frac{|\text{Wigner’s Friend}\rangle |\text{Wigner}_0\rangle + |\text{Wigner’s Friend}\rangle |\text{Wigner}_1\rangle}{\sqrt{2}} \end{aligned}$$

From Wigner’s point of view, he’s thinking one thought or the other one. But, from his friend’s point of view, Wigner isn’t thinking either of them until a measurement gets made. After Wigner’s friend makes a measurement, the state of his own mind will change to either a state in which he saw that Wigner wanted pancakes or one where he saw Wigner wanted eggs. At that point we’ll have an entangled state like

$$\frac{|\text{Wigner’s Friend}_0\rangle |\text{Wigner}_0\rangle + |\text{Wigner’s Friend}_1\rangle |\text{Wigner}_1\rangle}{\sqrt{2}}$$

But then what happens if another friend comes along, and then another? The point is to highlight an apparent incompatibility between the perspectives of different observers. It seems like either we need to retreat into a sort of solipsism—holding that an event that happened for Wigner might not have happened for his friend—or else we need some way of regarding the measurement as fictitious.

12.3 Dynamical Collapse

If quantum mechanics doesn’t make sense to us, it’s worth at least considering the possibility that it’s not a complete theory. That is, maybe it does a good job of describing microscopic systems but there are additional unaccounted-for rules needed to describe reality as a whole. Perhaps there may be some physics that we haven’t discovered yet which would show that qubits normally evolve via unitary transformations, but that for sufficiently large systems the

superposition states tend to spontaneously collapse to some classical state. In that case, we could view collapse as a straightforward physical process that turns pure states into mixed states. Theories which posit the existence of such an undiscovered collapse mechanism are generally referred to as **Dynamical Collapse Theories**. In order to make contact with existing quantum theory, dynamical collapse theories generally suppose that the new mechanism has the effect of physically instantiating the collapse of superposition states to classical outcomes with probabilities given by the Born rule. That is,

$$\sum_i \alpha_i |i\rangle \rightarrow |i\rangle \quad \text{with probability } |\alpha_i|^2$$

In the Schrödinger’s cat example, dynamical collapse theories would say that it doesn’t matter how isolated the box is, there is some yet-unknown physical law that says that a system that big would quickly evolve into a mixed state.

$$\frac{|\text{🐱}\rangle + |\text{🐱}\rangle}{\sqrt{2}} \rightarrow \frac{|\text{🐱}\rangle \langle \text{🐱}| + |\text{🐱}\rangle \langle \text{🐱}|}{2}$$

Note that in principle, there’s a measurement that can distinguish the two states above.

Such a measurement would admittedly be absurdly hard to implement. In fact, a recent result by Prof. Aaronson says informally that if you have the technological capability to distinguish the two states above, then you also have the technological capability to rotate between the cat’s “alive” and “dead” states. For this reason, the Schrödinger’s cat experiment involves far less animal cruelty than most people say! If you can do the experiment at all and prove that you did it, then you can also bring a dead cat back to life!

Setting aside technological difficulties, for us the relevant point is that in saying that a superposition state can evolve to a mixed state we’re necessarily proposing new physics. This prediction is different from what we’d find using standard quantum mechanics and so in principle has testable implications. In other words, this isn’t really interpreting quantum mechanics so much as it’s proposing a rival theory! Physicists have a high bar for such proposals; the burden of proof is on the person proposing the new law to explain in quantitative detail how it works. In this case that would mean giving a criterion for exactly which systems are “big” enough, or whatever, to trigger a collapse like

the above—ideally deriving that criterion from more fundamental laws. Some suggestions include:

- ▶ Collapse happens when some number of atoms get involved.
- ▶ Collapse happens after a certain total mass is reached.
- ▶ Collapse happens when a system reaches a certain level of “complexity.”

On their face all these views seem contradictory to our understanding of physics which relies on reductionism; each atom’s dynamics obey the same set of simple equations regardless of how big or complicated a system the atom might be part of.

12.3.1 Ghirardi-Rimini-Weber (GRW) Theory

Ghirardi-Rimini-Weber (GRW) Theory avoids the contradiction with reductionist principles by positing that each atom has some tiny probability of collapsing at each point in time. For a system like Schrödinger’s cat, we only need one atom to collapse in order to cause the entire cat to collapse to the “alive” or “dead” states. This is consistent with the usual partial measurement rule of quantum mechanics. By analogy, measuring just one qubit of

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

will cause all of the qubits to collapse to either $|0 \cdots 0\rangle$ or $|1 \cdots 1\rangle$. While the probability for some given atom to collapse is minuscule, with a sufficiently large number of atoms in our system (say $\sim 10^{23}$ for a typical cat) the probability that *at least one* of the atoms has collapsed can be overwhelming. In the GRW proposal, macroscopically large superposition states (often called cat-states) are inherently unstable—the bigger the system, the shorter expected lifetime of maintaining a Schrödinger-cat-like state.

12.3.2 Penrose Theory

Penrose Theory is an alternative approach to dynamical collapse which says that superpositions spontaneously collapse when the superposed objects have a sufficiently large mass and are separated by a sufficiently large distance. *Why mass and distance?* Say we have a large massive object in a superposition of two different spatial locations:

$$\frac{1}{\sqrt{2}} (|\text{mountain} \text{ : } \rangle + | \text{ : } \text{mountain} \rangle).$$

General relativity tells us that mass curves nearby space-time in a manner analogous to a weight deforming a mattress. This means that a mass in one location would make spacetime curve differently than the same mass in a different location. The thing is, no one really knows how to combine general relativity and quantum mechanics; it's one of the biggest unsolved problems in physics. Ordinary quantum mechanics presupposes a fixed space and time, or at the very least fixed causal structure. In terms of quantum circuits, if you like, we require the ability to assign to a collection of qubits some definite order to the gates acting on those qubits. No one quite knows what it means to have a quantum superposition of different causal structures—yet, that seems to be what we'd be talking about in the situation with the widely-separated masses. So, Penrose's proposal is basically that this could be the place where quantum mechanics breaks down and is superseded by a more complete theory that includes gravitationally-induced “spontaneous collapses.”

Penrose then has further ideas about how all of this might be related to consciousness, which we won't go into.

One difficulty with dynamical collapse theories is that experimental groups have continued to produce examples of larger and larger states in superposition. As long as that continues, it seems like the believers in these theories will always need to be on the defensive—adjusting their answers to questions like “how much mass is enough to collapse a state?” to avoid contradicting the latest experiments. Early in this course we discussed the significance of the double-slit experiment performed with photons. In 1999 the Zeilinger group at the University of Vienna managed to perform the double-slit experiment with buckyballs (molecules with 60 carbon atoms and hundreds of electrons). In 2019 a collaborative effort between experimentalists at the University of Vienna and the University of Basel successfully performed the experiment with large organic molecules consisting of ~ 2000 atoms.

Superconducting Circuits

Beyond the double-slit experiment, another quantum phenomenon which is routinely used today to perform experiments with a macroscopically large number of particles is superconductivity. If you take a metallic coil (made of aluminum, for example) that is perhaps a micrometer across and cool it to nearly absolute zero, it is possible to induce a dissipationless current flow in a superposition of clockwise or counterclockwise around the coil. A picture of such a loop can be found in Figure 12.1. The number of electrons constituting

this current can number in the billions or trillions and so this is an example of a quantum superposition involving billions of particles!

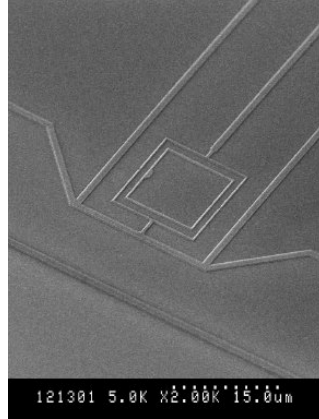


Figure 12.1: Scanning electron microscope (SEM) image of a superconducting flux qubit fabricated at the NTT Basic Research Lab in Japan, from DOI:10.1142/9789812774705_0003. When cooled to near absolute zero temperature the current in the loop can be in a superposition of circulating clockwise and counter-clockwise.

We'll come back to these superconducting coils at the end of the course as they're an important technology for quantum computers.

Limitations of Dynamical Collapse

Penrose made a specific prediction for the mass scale at which collapse happens—about 1 Planck mass, which is roughly the mass of a dust mite—which might be testable in our lifetime. But, with GRW the prediction is basically just made to avoid contradicting any existing experiments.

One position, popular among quantum computing skeptics, says that:

Perhaps a frog can be in a superposition of two states. However, a complex quantum computer wouldn't work because quantum systems spontaneously collapse after they achieve "sufficient complexity" (whatever that means).

This position is interesting because it could be falsified by building a scalable quantum computer. Making potentially falsifiable claims is what enables us to move these discussions from the realm of philosophy to science.

But what happens if we keep doing experiments and quantum mechanics keeps perfectly describing everything we see? In particular, suppose we don't want to add any new physical laws but, we also insist on being scientific realists—holding that there exists a real state of the universe and that the job of physics is to describe that state, not just to predict the results of measurements. The next interpretation is an attempt to satisfy those constraints by getting rid of a fundamental role for measurement altogether.

12.4 The Many-Worlds Interpretation

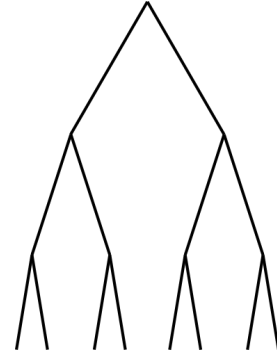
In 1957 Hugh Everett III, then a PhD student working under John Archibald Wheeler at Princeton, proposed a new interpretation of quantum mechanics he called the “relative state formulation.” We now refer to this interpretation as the **Many-Worlds Interpretation (MWI)** of quantum mechanics. This famous view holds that the entire universe has a single quantum state $|\psi\rangle$ and the entire history of the universe is just the result of the vector $|\psi\rangle$ undergoing unitary evolution.

In the MWI what we call “measurement” or “collapse” is a result of quantum systems becoming entangled with each other when they interact. In particular, your brain—not to mention your measuring apparatus, the air molecules in the room, etc. . . —all become entangled with the quantum system that you're measuring. You can think of it as a giant CNOT gate with the system you're observing as the control qubit and you and the environment as the target qubit. This situation is analogous to what we saw in the Wigner's friend thought experiment.

$$\frac{|0\rangle + |1\rangle}{\sqrt{2}} |\text{You}\rangle \rightarrow \frac{|0\rangle |\text{You}_0\rangle + |1\rangle |\text{You}_1\rangle}{\sqrt{2}}$$

The thing is, if you take this view seriously it implies that you yourself have now “branched” into two possibilities, one where you observed the qubit in the $|0\rangle$ state and another where you observed the qubit in the $|1\rangle$ state. Because unitary evolution is linear, these two branches are unlikely ever to interfere with each other again, or at least not for many quadrillions of years (more about that later). So, we might imagine, your experience is *as if* only one of the branches was realized—but, in truth neither branch is more real than the other.

More generally, according to MWI the universe “branches” each and every time a microscopic quantum state gets amplified to have macroscopic effect—even if there’s no one around to observe the amplification (e.g., if it’s in the interior of the Sun or something). There’s a staggering amount of branching happening! So for example, there are some branches of the quantum state of the universe where Austin is sunny at a specific moment one month from now, others where it’s rainy, others where it’s been destroyed in a nuclear war, etc., and no one of these branches is more “real” than the rest.



Some variants of the MWI choose words carefully to avoid sounding like there’s literal branching into different, equally-real worlds of experience, but that’s basically what they all imply. In fact, when Everett first came up with MWI as a grad student at Princeton, John Wheeler told him to remove from his paper all references to the physical reality of parallel worlds, because Bohr and his friends had strenuously objected to the idea. Everett did so, and partly as a result it took 15 years for the rest of the physics community to rediscover Everett’s proposal and understand what it meant.

After publishing his thesis about MWI Everett left theoretical physics to become a nuclear war strategist for the Pentagon. One of the only public lectures he ever gave on MWI was here at UT Austin, decades later, when people were finally coming around to the idea. David Deutsch, the biggest current advocate of the MWI and one of the founders of quantum computing, was there.

One issue that we should return to is interference between branches of the wavefunction. If different branches could interfere with each other, it would be as if not just the future but the past was constantly shifting, with destructively interfering paths (and their respective histories) having fleeting existences. There would be no definite history of things that happened and were recorded. To avoid this we need the $|0\rangle |You_0\rangle$ branch to not affect the $|1\rangle |You_1\rangle$ branch and vice versa. Both branches might be equally real, but once you’re in one of the branches you ought to be able to continue doing physics as if your branch was the only real one. Fortunately, the usual rules of quantum mechanics ensure this is the case; we don’t need to add anything extra. Recall that to calculate the amplitude of a given basis state following some sequence

of unitary transformations, you add up a contribution from every possible path that ends at that state. We saw an example of this in process in Figure 3.2. Interference happens only if two different paths lead to exactly the same outcome—meaning every single atom in the universe in the *exact* same state. While that’s not impossible, it’s massively “thermodynamically disfavored,” which basically means that it’s unlikely to happen for the same reason an egg unscrambling itself is unlikely to happen. In fact, the constant proliferation of branches—the way the universe’s state, $|\psi\rangle$, constantly sprouts new branches but almost never recombines them—can be seen as literally an instance of the Second Law of Thermodynamics in action, a process that constantly increases the universe’s entropy.

Interestingly, if we also believed that the universe was only finitely large—and in particular that it could be fully described by the unitary evolution of a finite number of qubits (say 10^{122} of them)—then eventually we’d run out of room and the branches would necessarily start colliding with each other. Even under that assumption, there doesn’t seem to be any reason for this to happen even in (say) the next 10^{100} years.

We said before that measurement is the one random and irreversible part of quantum mechanics. However, the MWI denies that even that part is random or irreversible. After applying a unitary transformation U which induces a branching corresponding to a “measurement”, we could in principle always apply U^\dagger to unperform the “measurement.” Just like with unscrambling an egg though, thermodynamics isn’t going to make it easy.

Let’s now discuss some of the most common other questions people have about Many Worlds.

Even if we accept that “measurements” have no fundamental physical status—still, where do the apparent probabilities come from? That is, why does measuring a qubit $\alpha|0\rangle + \beta|1\rangle$ in the $\{|0\rangle, |1\rangle\}$ basis yield the outcomes $|0\rangle$ and $|1\rangle$ with probabilities $|\alpha|^2$ and $|\beta|^2$ respectively?

It’s not enough to say that sometimes we see $|0\rangle$ and sometimes we see $|1\rangle$, quantum mechanics gives very specific probabilities for each of these events to occur. If the world is just branching once for each observation, then how can we justify these probabilities as corresponding to anything meaningful? Does an “ $|\alpha|^2$ fraction of your soul” go down one branch while a “ $|\beta|^2$ fraction of your soul” goes down the other?

Some philosophers don’t like this because if all the worlds are equally real, then why wouldn’t they just occur with equal probabilities? Why bother with amplitudes at all? Everett’s response was to argue that if the universe branched many times in succession then in “almost all branches” (where “almost all” is measured by amplitude) it would look like the Born probability

rule was obeyed. But, many people in the past half-century have been unsatisfied with that argument, seeing it as circular—as it essentially smuggles the Born rule into the definition of “almost all branches”! So, proponents have continued to look for something better.

There are many arguments, which we won’t go into here, that try to formalize the intuition that the Born probabilities are naturally “baked into” how quantum mechanics works. After all, unitary evolution already singles out the 2-norm as special by preserving it, so then why shouldn’t the probabilities also be governed by the 2-norm? More pointedly, one can argue that, if the probabilities were governed by something other than the 2-norm, then we’d get bizarre effects like faster-than-light communication. But, while these arguments help explain why the Born rule is perhaps the only choice of probability rule that makes internal mathematical sense, they still leave slightly mysterious how probability enters at all into Everett’s vision of a deterministically evolving wavefunction. In Everett’s defense, one could ask the same questions—*where do these probabilities come from? why should they follow the Born rule, rather than some other rule?*—in any interpretation, not just in MWI.

If there’s no experiment that could differentiate the Copenhagen Interpretation from Many Worlds, why bother arguing about it?

Many Worlders say that the opponents of Galileo and Copernicus could also claim the same about the Copernican versus Ptolemaic theories, since Copernican heliocentrism made no difference to the predictions of celestial movement. Today we might say that the Copernican view is better since if you were to fly outside of the solar system and see all the planets (including Earth) rotating around the far more massive sun, you’d realize that the Copernican was closer to reality; it’s only our parochial situation of living on Earth that ever motivated geocentrism in the first place. If we push this analogy further, it might be harder to think of anything similar for the Many Worlds interpretation, since quantum mechanics itself explains why we can’t really get outside of the state $|\psi\rangle$ to see the branching—or even get outside our own branch to interact in any way with the other decoherent branches.

There is one neat way you could imagine differentiating the two, though. Before we talked about doing the double-slit experiment with larger and larger systems. Bringing that thread to its logical conclusion, *what if we could run the double-slit experiment with a person going through the slits?* It seems like it would then be necessary to say that “observers” can indeed exist in superpositions of having one experience and having a different one. This is what the MWI said all along, but it seems to put a lot of rhetorical strain on the Copenhagen interpretation. If you talk to modern Copenhagenists about

this they'll often take a quasi-solipsistic view, saying that if this experiment were run “the person behaving quantumly doesn't count as an observer; only I, the experimenter, do.” Of course, the Wigner's Friend thought experiment was trying to get at this same difficulty.

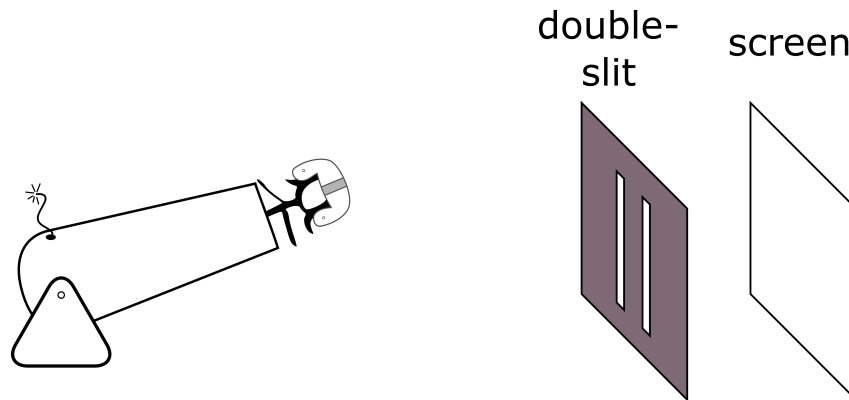


Figure 12.2: Potential experimental design for double-slit experiment using people. Best of luck to the future physicists who need to get this past their university's IRB.

Let's say I buy into the argument that the universe keeps branching. In what basis is this branching occurring?

This third question is called the **Preferred Basis Problem**. We talked about Schrödinger's cat as branching into the $|\text{😺}\rangle$ state and the $|\text{😿}\rangle$ state. But mathematically we could equally well have decomposed the cat's state in a basis like:

$$\frac{|\text{😺}\rangle + |\text{😿}\rangle}{\sqrt{2}} \quad \frac{|\text{😺}\rangle - |\text{😿}\rangle}{\sqrt{2}}$$

So, is there anything besides our intuition to “prefer” the first decomposition over the second one? There's a whole field of physics that tries to answer questions like these, called **Decoherence Theory**. The central idea is that there are certain bases whose states tend to be robust to interactions with the environment; most bases, however, don't have this property. In the example above, decoherence theory would explain that an alive cat doesn't easily decohere if you poke it, but a cat in the $\frac{1}{\sqrt{2}}(|\text{😺}\rangle + |\text{😿}\rangle)$ state does, because the $|\text{😺}\rangle$ and $|\text{😿}\rangle$ branches interact differently with the environment. This, according to decoherence theory, is more-or-less how the laws of physics pick out certain bases as being special.

From the standpoint of decoherence theory we can say that an event has “definitely happened” only if there exist many records of the event spread through the environment, so that it’s no longer feasible to erase them all.

This is perhaps best compared to putting an embarrassing picture on Facebook. If only a few friends share it, you can still take it down. On the other hand, if the picture goes viral, then the cat is out of the bag and deleting all the copies becomes next to impossible.

Lecture 13: Hidden Variables and Bell's Inequality

13.1 Hidden Variable Theories

In the last lecture we discussed four different attitudes people take toward quantum mechanics: Copenhagen, “Shut Up and Calculate,” Dynamical Collapse, and Everett’s Many-Worlds Interpretation. You might think that *all* the options we’ve seen so far are bizarre and incomprehensible, and wonder if we could come up with a theory that avoids all of the craziness. One strategy for the development of such a theory is the introduction of hidden variables in the system. These hidden variables are typically additional degrees of freedom which we either don’t have access to or haven’t yet discovered. The resulting theories are called **Hidden Variable Theories**.

Hidden variable theories supplement quantum state vectors with the addition of hidden ingredients. The idea is to have a state, like $\alpha |0\rangle + \beta |1\rangle$, represent “merely” a way of making a prediction about what the universe *has already* set the result of measuring the qubit to be: either $|0\rangle$ or $|1\rangle$.

13.1.1 Bohmian Mechanics

The most famous hidden-variable theory is **Bohmian Mechanics**, which was developed by David Bohm in the 1950s. It’s also called the “de Broglie-Bohm theory,” because it turns out that Louis de Broglie had the exact same idea in the 1920s—although de Broglie quickly disavowed it after the idea faced a poor reception from other quantum mechanics pioneers.

Normal quantum mechanics says that a particle is in a superposition of locations, which we can use to calculate the probability that the particle will be found in one place or another when measured—and moreover, that this superposition exhausts what can be said about the particle’s location. But, while keeping that superposition as part of physics, we now want to say that there’s

also a “real place” where the particle is, even before anyone measures it. To make that work we need to give a rule for how the superposition “guides” the real particle. This rule should have the property that, if anyone *does* measure the particle, they’ll find exactly the result that quantum mechanics predicted for it—since we certainly don’t want to give up on quantum mechanics’ empirical success!

At first, you might think that it would be tricky to find such a rule; indeed, you might wonder whether such a rule is possible at all. However, the real problem turns out to be more like an embarrassment of riches! There are infinitely many possible rules that could satisfy the above property—and by design, they all yield exactly the same predictions as standard quantum mechanics. So there’s no experimental way to know which one is correct.

To explain this in a bit more detail, let’s switch from particle positions back to the discrete quantum mechanics that we’re more comfortable with in this course. Suppose we have a quantum pure state, represented as an amplitude vector in some fixed basis. Then when we multiply by a unitary transformation, suppose we want to be able to say: “this is the basis state we were *really* in before the unitary was applied, and this is the one we’re really in afterwards.” In other words, we want to take the equation

$$\begin{bmatrix} \beta_0 \\ \vdots \\ \beta_{n-1} \end{bmatrix} = \begin{bmatrix} U_{0,1} & \cdots & U_{0,n-1} \\ \vdots & \ddots & \\ U_{n-1,1} & & U_{n-1,n-1} \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{n-1} \end{bmatrix} \quad (13.1)$$

and map it to an equation

$$\begin{bmatrix} |\beta_0|^2 \\ \vdots \\ |\beta_{n-1}|^2 \end{bmatrix} = \begin{bmatrix} S \end{bmatrix} \begin{bmatrix} |\alpha_0|^2 \\ \vdots \\ |\alpha_{n-1}|^2 \end{bmatrix} \quad (13.2)$$

for some choice of stochastic matrix S (possibly depending on the input and output vectors). There are many, many such matrices S . For example, you could put $[|\beta_0|^2, \dots, |\beta_{n-1}|^2]^\top$ in every column, which would say that you’re always jumping randomly around, but in a way that preserves the Born rule. You could have been in a different galaxy one Planck time ($\sim 10^{-43}$ seconds) ago; now you’re here (with fictitious memories planted in your brain); who knows where you’ll be a Planck time from now?

Bohm, however, thought not about this discrete setting, but mostly about the example of a particle moving around in continuous Euclidean space. In the latter case it turns out that one can do something nice that isn’t possible

with finite-dimensional unitary matrices. Namely, one can give a *deterministic* rule for how the particle moves around—a differential equation—that still reproduces the Born rule at every instant in time, provided only that it reproduces the Born rule at any *one* time. More poetically, “God needs to use a random-number generator to initialize the hidden variables at the beginning of time”—say, at the Big Bang—but afterwards, they just follow the differential equation. Furthermore, while the choice of differential equation isn’t quite unique, in simple scenarios (like a particle moving around in space) there’s one choice that seems better, simpler, and more motivated than the rest.

However, in thinking through the implications of Bohmian mechanics, Bohm and others noticed lots of weird things. It looks very elegant with just one particle, but new issues arise when there are two entangled particles. Bohmian mechanics says that you need to give a definite position for both particles, but people noticed that acting on Alice’s particle would *instantaneously* change the Bohmian position of Bob’s particle, however far away the particles were—even while Bob’s density matrix remained unchanged because of the No-Communication Theorem. Because of these instantaneous changes in the Bohmian position, we refer to Bohmian mechanics as a **Nonlocal Hidden Variable Theory**.

While unsettling, this still wouldn’t be useful for faster-than-light communication, since the Bohmian hidden variables are explicitly designed to have no measurable effects beyond the effects we’d predict using the quantum state itself.

When Bohm proposed his interpretation, he was super eager for Einstein (whose objections to quantum mechanics we’ve discussed previously) to accept it, but Einstein didn’t really go for it, probably because of this nonlocality problem. What Einstein really seems to have wanted (in modern terms) is a **Local Hidden Variable Theory** where hidden variables not only exist, but can be localized to specific points in space and are only influenced by things happening close to them.

13.1.2 Local Hidden Variable Theories

Imagine that when an entangled pair, $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$, is created the qubits secretly flip a coin and decide, “if anyone measures us in the $\{|0\rangle, |1\rangle\}$ basis let’s both be $|0\rangle$.” More broadly, imagine that they agree in advance on such answers for all questions that could be asked (i.e., all bases in which they could possibly

be measured), and that each qubit carries around its own local copy of the answers.

This is *not* Bohmian mechanics. In fact, around 1963 John Bell wrote a paper that drew attention to the nonlocal character of Bohmian mechanics. Bell remarked that it would be interesting to prove that *all* hidden variable theories must be nonlocal. In other words that nonlocality isn't just some defect of Bohm's proposal, but inherent to hidden variable theories in general. The paper has a footnote saying that as the paper was going to press, such a proof was found. This was the first announcement of one of the most famous discoveries ever made about quantum mechanics, what we now call **Bell's Theorem**.

Einstein and others had already touched on the idea of local hidden variable theories in their philosophical debates in the 1930s. Bell was the first to ask: *do local hidden variables have any empirical consequences that disagree with the predictions of quantum mechanics? Is there an actual experiment that could rule out the possibility of local hidden variables?* Bell came up with such an experiment. We'll describe it differently from how Bell did originally—more computer sciencey—as a game with two cooperating players named (what else?) Alice and Bob, where the achievable win probability can be improved through shared entanglement (to a value higher than is possible classically). This game is called the **CHSH Game**.

13.2 The CHSH Game

The **CHSH Game** is named after four people (Clauser, Horne, Shimony, and Holt) who in 1969 wrote a paper saying “*this* is how to think about what Bell did.” The game itself doesn't involve quantum mechanics, but quantum mechanics can help us win it.

The CHSH game could be seen as a precursor to quantum computing, in that it's one of the first cases where people looked to see which information processing tasks quantum mechanics helps us solve better—and where they enforced a conceptual separation between the task itself (which is classical) and the strategy to solve it (which can be quantum).

The idea is that Alice and Bob are placed in separate rooms and are both given a challenge bit (x and y , respectively) by a referee, Charlie. The challenge bits are chosen uniformly at random, and independently of each other. Alice

sends an answer bit, a , back to the referee and Bob sends back an answer bit b . Alice and Bob “win” the game iff

$$a + b = xy \pmod{2}. \quad (13.3)$$

Alice and Bob are allowed to agree on a strategy in advance (in other words correlate their answers) and to share random bits. This situation is shown in Figure 13.1.

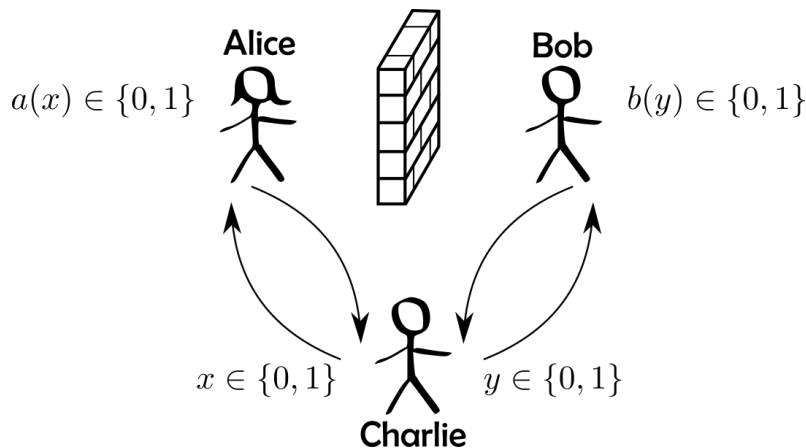


Figure 13.1: Diagrammatic depiction of the CHSH game. Charlie, the referee, prepares two challenge bits x and y uniformly at random and sends them to Alice and Bob respectively. Alice and Bob in response send bits a and b respectively (which could depend on their inputs) back to Charlie with the goal of having $a + b = xy \pmod{2}$. In other words, Alice and Bob want to select bits such that the parity of their selected bits is equal to the AND of their input bits.

A classical strategy to maximize winning probability is simply that Alice and Bob always send the referee $a = b = 0$, regardless of what x and y are. In this case, Alice and Bob win 75% of the time, losing only if x and y are both 1. To prove that this is optimal the first step is to notice that, without loss of generality, Alice and Bob’s strategy can be assumed to be deterministic (i.e., to involve no random bits besides x and y themselves). Any probabilistic strategy is equivalent to a mixture of deterministic ones. As such, the win probability for a probabilistic strategy is the weighted average over all the deterministic strategies the probabilistic strategy employs. There must be some deterministic strategy in the mixture that does at least as well as the average, so we can derandomize the protocol without negatively impacting the overall success probability; this is called a convexity argument. This convexity

argument holds true even if we assume that Alice and Bob have access to shared randomness.

Let's treat Alice's output bit a as a function of her input bit x and Bob's output bit b as a function of his input bit y . In order to win the game they need to select bits a and b satisfying the equation

$$a(x) + b(y) = xy \pmod{2} . \tag{13.4}$$

You can easily check by enumerating cases (as we do in Table 13.1) that this equation can't possibly hold for all 4 values of x and y ! At best it can hold for 3 of the 4 values, which is exactly what the trivial strategy above (always send back 0) achieves.

Strategy	x	y	a	b	a+b	xy
Always Send 0	0	0	0	0	0	0
	0	1	0	0	0	0
	1	0	0	0	0	0
	1	1	0	0	0	1
Always Send 1	0	0	1	1	0	0
	0	1	1	1	0	0
	1	0	1	1	0	0
	1	1	1	1	0	1
Same as Input	0	0	0	0	0	0
	0	1	0	1	1	0
	1	0	1	0	1	0
	1	1	1	1	0	1
Opposite of Input	0	0	1	1	0	0
	0	1	1	0	1	0
	1	0	0	1	1	0
	1	1	0	0	0	1

Table 13.1: Enumeration of all possible deterministic strategies for the CHSH game. Lines where $a + b \neq xy \pmod{2}$ are colored red.

The **Bell Inequality**, in this framework, is just the slightly boring statement that we proved above. Namely, that the maximum classical win probability in the CHSH game is 75%. Bell noticed an additional fact, though. If Alice and Bob have access to a pre-shared Bell pair, $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$, then there's a better strategy. In that case, in fact, their maximum win probability is

$$P = \cos^2\left(\frac{\pi}{8}\right) \approx 85\%$$

How do they use entanglement to achieve an improvement over the classical win probability? Tune in next time to find out!

Lecture 14: Nonlocal Games

14.1 CHSH Game: Quantum Strategy

In the previous lecture we talked about the CHSH Game, and how no classical strategy lets Alice and Bob win it more than 75% of the time. In this lecture we'll see how, by using entanglement, they can win $\approx 85\%$ of the time, and delve deeper to try to understand what's going on.

The quantum strategy for the CHSH game assumes that before Alice and Bob go to their separate rooms and receive their challenges they pre-share 1 ebit of entanglement. In other words Alice and Bob each share 1 half of a Bell pair, $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$. Upon receiving their challenge bits from the referee, Charlie, Alice and Bob then measure their respective qubits in different bases depending on whether their input bits x and y are 0 or 1. Based on the results of their measurements, Alice and Bob then output bits a and b based on the outcomes.

Before explicitly stating the protocol we'll first quickly introduce notation for four states which will play a crucial roll in the protocol. These four states are

$$|\pi/8\rangle = \cos\left(\frac{\pi}{8}\right)|0\rangle + \sin\left(\frac{\pi}{8}\right)|1\rangle, \quad |5\pi/8\rangle = \cos\left(\frac{5\pi}{8}\right)|0\rangle + \sin\left(\frac{5\pi}{8}\right)|1\rangle \quad (14.1)$$

$$|-\pi/8\rangle = \cos\left(-\frac{\pi}{8}\right)|0\rangle + \sin\left(-\frac{\pi}{8}\right)|1\rangle, \quad |3\pi/8\rangle = \cos\left(\frac{3\pi}{8}\right)|0\rangle + \sin\left(\frac{3\pi}{8}\right)|1\rangle \quad (14.2)$$

The states correspond to a rotation of the standard basis either $\pi/8$ radians counterclockwise, as in Equation 14.1, or $\pi/8$ radians clockwise, as in Equation 14.1. We actually saw at least one of these states in a different context back in Section 5.2 when discussing the distinguishability of quantum states—though

we may not have written it explicitly in this form—and on the homework in calculating the optimal basis for distinguishing the $|0\rangle$ state from the $|+\rangle$ state. Recall that measuring in a basis $\{|v\rangle, |w\rangle\}$ is equivalent to applying a unitary transformation into the standard basis followed by a measurement in the standard basis. As such, we can perform measurements in the $\{|\pi/8\rangle, |5\pi/8\rangle\}$ and $\{|-\pi/8\rangle, |3\pi/8\rangle\}$ bases by applying unitary transformations (in this case $R_{\pi/8}$ and $R_{-\pi/8}$, respectively) followed by a measurement in the standard basis.

Alice and Bob's measurement strategy is as follows:

- ▶ If $x = 0$ then Alice measures in the $\{|0\rangle, |1\rangle\}$ basis. If the outcome of Alice's measurement is $|0\rangle$, then she sends back $a = 0$. If the outcome of Alice's measurement is $|1\rangle$, then she sends back $a = 1$.
- ▶ If $x = 1$ then Alice measures in the $\{|+\rangle, |-\rangle\}$ basis. If the outcome of Alice's measurement is $|+\rangle$, then she sends back $a = 0$. If the outcome of Alice's measurement is $|-\rangle$, then she sends back $a = 1$.
- ▶ If $y = 0$ then Bob measures in the $\{|\pi/8\rangle, |5\pi/8\rangle\}$ basis. If the outcome of Bob's measurement is $|\pi/8\rangle$, then he sends back $b = 0$. If the outcome of Bob's measurement is $|5\pi/8\rangle$, then he sends back $b = 1$.
- ▶ If $y = 1$ then Bob measures in the $\{|-\pi/8\rangle, |3\pi/8\rangle\}$ basis. If the outcome of Bob's measurement is $|-\pi/8\rangle$, then he sends back $b = 0$. If the outcome of Bob's measurement is $|3\pi/8\rangle$, then he sends back $b = 1$.

This strategy has the amazing property that Alice and Bob win with probability $\cos^2(\pi/8)$ for all possible values of x and y .

14.1.1 Analysis of Protocol

So why does this strategy work 85% of the time? Let's analyze the results on a case-by-case basis. We can assume without loss of generality that Alice measures first, due to the No-Communication Theorem. The first case is where Alice gets $x = 0$ and Bob gets $y = 0$. As always, Alice and Bob will win if they return a and b such that $a + b = xy = 0 \pmod{2}$. Alice measures in the $\{|0\rangle, |1\rangle\}$ basis and Bob measures in the $\{|\pi/8\rangle, |5\pi/8\rangle\}$ basis. Suppose Alice sees $|0\rangle$. Then the state of Bob's qubit also collapses to $|0\rangle$. When Bob measures his now collapsed qubit in the $\{|\pi/8\rangle, |5\pi/8\rangle\}$ basis he sees either $|\pi/8\rangle$ or $|5\pi/8\rangle$ with probabilities

$$\begin{aligned} P(|\pi/8\rangle) &= |\langle 0|\pi/8\rangle|^2 = |\cos(\pi/8)|^2 = \cos^2\left(\frac{\pi}{8}\right) \\ P(|5\pi/8\rangle) &= |\langle 0|5\pi/8\rangle|^2 = |\cos(5\pi/8)|^2 = |\sin(\pi/8)|^2 = \sin^2\left(\frac{\pi}{8}\right) \end{aligned} \tag{14.3}$$

Now suppose Alice sees $|1\rangle$. Then Bob's qubit collapses to $|1\rangle$ and when he measures in the $\{|\pi/8\rangle, |5\pi/8\rangle\}$ basis he sees either $|\pi/8\rangle$ or $|5\pi/8\rangle$ with probabilities

$$\begin{aligned} P(|\pi/8\rangle) &= |\langle 1|\pi/8\rangle|^2 = |\sin(\pi/8)|^2 = \sin^2\left(\frac{\pi}{8}\right) \\ P(|5\pi/8\rangle) &= |\langle 1|5\pi/8\rangle|^2 = |\sin(5\pi/8)|^2 = |\cos(\pi/8)|^2 = \cos^2\left(\frac{\pi}{8}\right) \end{aligned} \quad (14.4)$$

Recall that the goal is to output a and b such that $a + b = xy \pmod{2}$. In this first case $xy = 0$ so Alice and Bob win if they return either $a = b = 0$ or $a = b = 1$. The probability they win is thus given by

$$\begin{aligned} &P(a = 0)P(b = 0|a = 0) + P(a = 1)P(b = 1|a = 1) \\ &= \frac{1}{2} \cos^2\left(\frac{\pi}{8}\right) + \frac{1}{2} \cos^2\left(\frac{\pi}{8}\right) \\ &= \cos^2\left(\frac{\pi}{8}\right) \end{aligned}$$

The analysis for the case where $x = 0$ and $y = 1$ and the case where $x = 1$ and $y = 0$ is very similar to the case above, so we'll leave it to the reader to verify that the protocol indeed succeeds with probability $\cos^2(\pi/8)$ in those cases. Somewhat more interesting is the case where $x = y = 1$. In order to analyze this case it is useful to note that the Bell state takes the same general form regardless of which basis we choose to right it in. In other words, $\frac{|00\rangle + |11\rangle}{\sqrt{2}} = \frac{|vv\rangle + |ww\rangle}{\sqrt{2}}$ for any orthonormal basis $\{|v\rangle, |w\rangle\}$. In this case we'll choose to write it as $\frac{|++\rangle + |--\rangle}{\sqrt{2}}$. Therefore, when Alice measures in the $\{|+\rangle, |-\rangle\}$ basis she sees both $|+\rangle$ and $|-\rangle$ with equal probability. In either case, Bob's qubit collapses to the same outcome. Suppose Alice sees $|+\rangle$. Then when Bob measures his qubit in the $\{|-\pi/8\rangle, |3\pi/8\rangle\}$ basis he sees $|-\pi/8\rangle$ with probability

$$P(|-\pi/8\rangle) = |\langle +|-\pi/8\rangle|^2 = \sin^2\left(\frac{\pi}{8}\right) \quad (14.5)$$

and he sees $|3\pi/8\rangle$ with probability

$$P(|3\pi/8\rangle) = 1 - P(|-\pi/8\rangle) = 1 - \sin^2\left(\frac{\pi}{8}\right) = \cos^2\left(\frac{\pi}{8}\right) \quad (14.6)$$

Likewise, when Alice sees $|-\rangle$, Bob sees either $|-\pi/8\rangle$ with probability

$$P(|-\pi/8\rangle) = |\langle -|-\pi/8\rangle|^2 = \cos^2\left(\frac{\pi}{8}\right) \quad (14.7)$$

or $|3\pi/8\rangle$ with probability

$$P(|3\pi/8\rangle) = 1 - P(|-\pi/8\rangle) = 1 - \cos^2\left(\frac{\pi}{8}\right) = \sin^2\left(\frac{\pi}{8}\right). \quad (14.8)$$

The win condition for Alice and Bob when $x = y = 1$ is for a and b to have odd parity. As such, the win probability is:

$$\begin{aligned} & P(a = 0)P(b = 1|a = 0) + P(a = 1)P(b = 0|a = 1) \\ &= \frac{1}{2} \cos^2\left(\frac{\pi}{8}\right) + \frac{1}{2} \cos^2\left(\frac{\pi}{8}\right) \\ &= \cos^2\left(\frac{\pi}{8}\right) \end{aligned}$$

So to sum up, in all four cases the quantum protocol achieves a win probability of $\cos^2\left(\frac{\pi}{8}\right) \approx 85\%$, higher than the classical win probability of 75%.

14.1.2 CHSH Game: Interpretations and Local Realism

How does this game relate to hidden-variable theories? Well, if all correlations between the qubits could be explained by stories like “if anyone asks, we’re both 0,” then we’d make a firm prediction that Alice and Bob can win the CHSH game with at most 75% probability (because that’s how well they can do by pre-sharing arbitrary amounts of classical randomness). This would be consistent with the principle of local realism, which we discussed in Lecture 1. So, if they play the game repeatedly and demonstrate that they can win more than 75% of the time, then we must conclude local realism doesn’t hold. Notice that nowhere in this argument did we ever need to presuppose that quantum mechanics is true.

Does Alice and Bob’s ability to succeed more than 75% of the time mean that they are communicating? Well, we know it’s not possible for either to send a signal to the other using entanglement by the No-Communication Theorem. One way to understand what’s going on is to work out Alice and Bob’s density matrices explicitly.

Bob’s initial local density matrix is

$$\begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$$

and after Alice measures it’s still

$$\begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}.$$

So in that sense, no signal has been communicated from Alice to Bob. Nevertheless, *if* you knew both Alice’s measurement and its outcome, then you could update Bob’s local density matrix to that of a pure state. That shouldn’t worry us though since, even classically, if you condition on what Alice sees then you can change your predictions for Bob.

Imagine a hierarchy of possibilities for what the universe allows. Classical Local Realism is at the bottom. Here you only ever need to use classical probability theory when you have incomplete information about physical systems, and also signals propagate at most at the speed of light. At the top of the hierarchy is the Faster-Than-Light (FTL) Science-Fiction Utopia, where Alice and Bob can communicate instantaneously, can travel faster than light, and so forth. People tend to believe that reality must be one or the other. So when they read pop-science articles about how classical local realism doesn’t hold, they think, “OK, then we must live in the FTL sci-fi utopia.” Instead, the truth—according to quantum mechanics—is somewhere in between these two, and the distinction is so subtle that perhaps no science-fiction writer would ever have had the imagination to invent it. We live in a world where there’s no classical local realism, but no faster-than-light communication either. Or, to put it another way, a purely classical simulation of our universe would have to include FTL communication, but our universe itself does not.

Maybe no science fiction writer ever came up with this possibility simply because it’s hard to think of a plot that requires Alice and Bob to win the CHSH game 85% of the time instead of 75%!

Indeed, experimental violations of the Bell inequality are a key piece of evidence that our world really is quantum and is not secretly classical behind the scenes; we know from Bell that the latter possibility would require FTL communication.

14.1.3 Tsirelson’s Inequality

So where is that $\cos^2\left(\frac{\pi}{8}\right)$ coming from anyways? It seems so arbitrary... It may seem like the $\cos^2\left(\frac{\pi}{8}\right)$ is simply coming from our particular approach to the problem. You may think that maybe if we came at it another way, we could use entanglement to win even more than 85% of the time, perhaps even 100% of the time. Surprisingly, the $\cos^2\left(\frac{\pi}{8}\right)$ win probability turns out to be optimal for quantum strategies, even if Alice and Bob share unlimited amounts of entanglement. This result is known as **Tsirelson’s Inequality** or

Tsirelson's Bound and was proved in 1980 by Boris Tsirelson.

It requires a *bit* too much machinery to give a complete proof of Tsirelson's Inequality here. Instead we'll convey the intuition by showing that among strategies "similar to the one we used," ours was the optimal one. Let's say that Alice has two angles, θ_0 and θ_1 , corresponding to the bases obtained by applying R_{θ_0} and R_{θ_1} to the elements of the standard basis. If Alice receives the input $x = 0$ then she measures her half of the Bell pair in the θ_0 basis, and if she receives the input $x = 1$ she measures in the θ_1 basis. Similarly, Bob has a pair of angles ϕ_0 and ϕ_1 , that correspond to his measurement bases when he receives $y = 0$ and $y = 1$, respectively. For the quantum strategy we discussed earlier, Alice's angles are $\theta_0 = 0$ and $\theta_1 = \pi/4$, and Bob's angles are $\phi_0 = \pi/8$ and $\phi_1 = -\pi/8$.

Following essentially the same case-by-case analysis of the win probability we did above, it can be shown that the overall win probability as a function of θ_0 , θ_1 , ϕ_0 and ϕ_1 is given by

$$P(\text{win}) = \frac{1}{4} [\cos^2(\theta_0 - \phi_0) + \cos^2(\theta_0 - \phi_1) + \cos^2(\theta_1 - \phi_0) + \sin^2(\theta_1 - \phi_1)]. \quad (14.9)$$

Each of the four input pairs has an equal chance of occurring. In the first three cases, Alice and Bob win iff they output the same bit. The probability of this is given by the squared inner product of corresponding measurement basis elements, which is equal to the squared cosine of the difference between their measurement angles. In the fourth case, Alice and Bob win iff they output different bits. As such, we take the squared sine of the difference between their measurement angles. Using the power-reduction identity from trigonometry, we can rewrite 14.9 as

$$P(\text{win}) = \frac{1}{2} + \frac{1}{8} [\cos(2(\theta_0 - \phi_0)) + \cos(2(\theta_0 - \phi_1)) + \cos(2(\theta_1 - \phi_0)) + \sin(2(\theta_1 - \phi_1))] \quad (14.10)$$

We can then get rid of the 2's inside the cosines by folding them into our original angles. It will be helpful to think of the cosines as the inner products between unit vectors. In that case, we can rewrite the above as

$$\begin{aligned} & \frac{1}{2} + \frac{1}{8} [u_0 \cdot v_0 + u_0 \cdot v_1 + u_1 \cdot v_0 - u_1 \cdot v_1] \\ &= \frac{1}{2} + \frac{1}{8} [u_0 \cdot (v_0 + v_1) + u_1 \cdot (v_0 - v_1)]. \end{aligned} \quad (14.11)$$

Since $u_0, u_1, v_0,$ and v_1 are all unit vectors, the above is upper-bounded by

$$\frac{1}{2} + \frac{1}{8} [u_0 \cdot (v_0 + v_1) + u_1 \cdot (v_0 - v_1)] \leq \frac{1}{2} + \frac{1}{8} [\|v_0 + v_1\| + \|v_0 - v_1\|]. \quad (14.12)$$

Using the parallelogram law ($2\|x\|^2 + 2\|y\|^2 = \|x + y\|^2 + \|x - y\|^2$), we get

$$\frac{1}{2} + \frac{1}{8} [\|v_0 + v_1\| + \|v_0 - v_1\|] = \frac{1}{2} + \frac{1}{8} \left[\|v_0 + v_1\| + \sqrt{4 - \|v_0 + v_1\|^2} \right]. \quad (14.13)$$

Next, since v_0 and v_1 are orthonormal, $\|v_0 + v_1\| = \sqrt{2}$. So we find

$$\begin{aligned} \frac{1}{2} + \frac{1}{8} [\sqrt{2} + \sqrt{2}] &= \frac{1}{2} + \frac{1}{8} [2\sqrt{2}] \\ &= \frac{1}{2} + \frac{1}{2\sqrt{2}} = \cos^2\left(\frac{\pi}{8}\right). \end{aligned} \quad (14.14)$$

So $\cos^2\left(\frac{\pi}{8}\right)$ really is the maximum winning probability for the CHSH game.

There's been a trend in the last 15 years to study theories that would go past quantum mechanics by letting you violate Tsirelson's Inequality, but that would still prohibit faster-than-light signals. In such a world, it's been proven (among other things) that if Alice and Bob wanted to schedule something on a calendar, they could decide if there's a date where they're both free by exchanging only one bit of communication. That's a lot better than can be done under the rules of quantum mechanics!

14.1.4 Experimental Tests of Bell's Inequalities

When Bell initially proved his inequality he was primarily trying to make a conceptual point about the need for nonlocal influences in any hidden-variable theory underlying quantum mechanics. But, by the 1980s, technology had advanced to the point where playing the CHSH game was actually a feasible physics experiment! Alain Aspect (and others) ran the experiment and the results were fully consistent with quantum mechanics, and extremely problematic for local hidden variable theories.

The experiments don't quite get to 85% success probability, given the usual difficulties that afflict quantum experiments. But, you can reach a high statistical confidence that you're winning more than, say, 80% of the time, well above the classical limit of 75%.

This was evidence, not only that local realism was false, but also that entanglement had been created. Most physicists shrugged, already sold on quantum mechanics (and on the existence of entanglement). But, a few, still committed to a classical view of the world, continued to look for loopholes in the experiment. Skeptics pointed out two main loopholes in the existing experiments, essentially saying “if you squint enough, classical local realism might still be possible”:

► Detection Loophole

- Sometimes detectors fail to detect a photon, or they detect non-existent photons (called “dark counts”). Enough noise in the experiments turns out to make a local hidden-variable explanation possible again.

► The Locality Loophole

- Performing the measurements and storing their results in a computer memory takes some time, maybe nanoseconds or microseconds. Now, unless Alice and Bob and the referee are very far away from each other, this opens the possibility of a sort of “local hidden variable conspiracy,” where as soon as Alice measures, some particle (unknown to present-day physics) flies over to Bob and says “hey, Alice got the measurement outcome 0, you should return the measurement outcome 0 too.” The particle would travel *only* at the speed of light, yet could still reach Bob before his computer registered the measurement outcome.

By the 2000s physicists were able to close the locality loophole, but only in experiments still subject to the detection loophole and vice versa. Finally, in 2016 several teams managed to do experiments that closed both loopholes simultaneously.

There are still people who deny the reality of quantum entanglement, but through increasingly solipsistic arguments. The last stronghold for these skeptics is the idea of **Superdeterminism**. Superdeterminism and the related “Freedom-of-Choice Loophole” explains the results of CHSH experiments by saying “we only think Alice and Bob can choose measurement bases randomly.

In actuality, there's a grand cosmic conspiracy involving all of our brains, our computers, and our random number generators, with the purpose of rigging the measurement bases to ensure that Alice and Bob can win the CHSH game 85% of the time. But that's all this cosmic conspiracy does! It doesn't allow FTL communication or anything like that, even though it easily could." Nobel Laureate Gerard 't Hooft (the 't is pronounced like "ut") advocates superdeterminism, so it's not like the idea lacks distinguished supporters, at the very least.

14.2 The Odd Cycle Game

Now we'll look at some other non-local games, to see what else entanglement can help with. First we'll look at the **Odd Cycle Game**. Suppose we have a cycle with an odd number of vertices n . Alice and Bob claim that they have a two-coloring of the cycle, but basic graph theory tells us that this isn't possible. Nevertheless, Alice and Bob make it their goal to try to convince a referee that they've found a two-coloring anyway. They'll do that by using entanglement to coordinate their responses to challenges from a referee.

The referee performs two obvious consistency checks:

- ▶ He can ask them both the color of a random vertex v , in the two-coloring they found.
 - They pass this test iff their answers are the same.
- ▶ He can ask Alice the color of a random vertex v , and Bob the color of an adjacent vertex w .
 - They pass the test iff their answers are different.

The referee chooses between these tests with equal probability—and crucially, he doesn't tell Alice or Bob which test he's performing. In a single run of the game, the referee performs one such test, and gets answers from Alice and Bob. We'll assume their answers are always RED or BLUE.

What strategy provides the best probability that Alice and Bob will pass the referee's test and win the game? Classically we know that, regardless of what Alice and Bob do, $P(\text{win}) < 1$. *Why?* One can show that for Alice and Bob to answer all possible challenges correctly they'd need an actual two-coloring, which is impossible. The best they can do is agree on a coloring for all but one of the vertices, which gives them a win probability of

$$P(\text{win}) = 1 - \frac{1}{2n}. \quad (14.15)$$

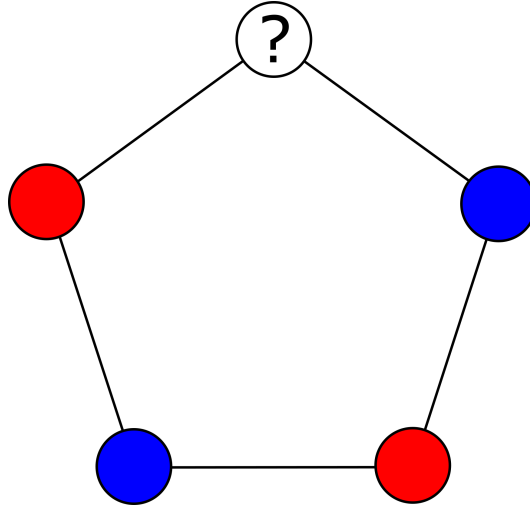


Figure 14.1: A 5-node cycle with a partial two-coloring. Notice that there is no consistent way to fill in a color for the node labeled “?”.

Nevertheless, we claim that quantum mechanically Alice and Bob can do better and can achieve

$$P(\text{win}) = 1 - \mathcal{O}\left(\frac{1}{n^2}\right) \quad (14.16)$$

if they pre-share a single Bell pair. The strategy is as follows: Alice and Bob both measure their qubits in a basis depending on the vertex they’re asked about. The measurement bases for adjacent vertices are rotated with respect to each other by $\frac{\pi}{2n}$. The first basis has outcome $|0\rangle$ map to answering BLUE and outcome $|1\rangle$ map to answering RED. The second basis has outcome $|\frac{\pi}{2n}\rangle$ map to RED and $|\frac{\pi}{2} + \frac{\pi}{2n}\rangle$ map to BLUE. They continue alternating this for the remaining bases. This scheme is sketched out in Figure 14.2.

When Alice and Bob are asked about the same vertex, they both measure in the same basis and thus always answer with the same color. On the other hand, when Alice and Bob are asked about adjacent vertices we get a similar situation to the CHSH game, where the probability of Bob producing the opposite output as Alice (and passing the test as a result) is the squared cosine of the angle between their measurement basis vectors. As such, the probability that they lose is

$$P(\text{lose}) = 1 - \cos^2\left(\frac{\pi}{2n}\right) = \sin^2\left(\frac{\pi}{2n}\right) \approx \left(\frac{\pi}{2n}\right)^2 = \mathcal{O}\left(\frac{1}{n^2}\right). \quad (14.17)$$

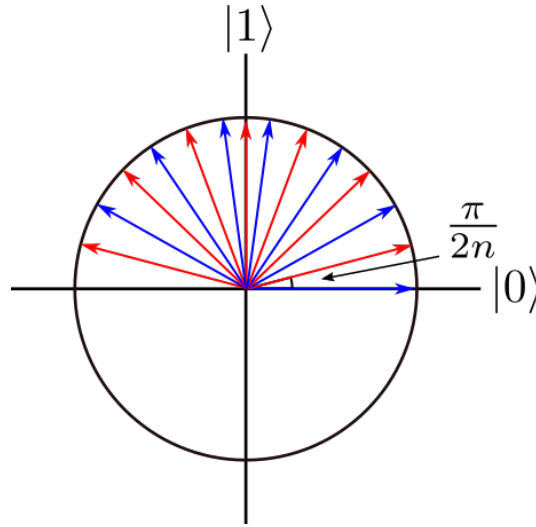


Figure 14.2: Diagrammatic depiction of the measurement bases and coloring convention used in the Odd Cycle game’s quantum strategy.

14.3 The Magic Square Game

In the **Magic Square Game** (also called the **Mermin-Peres Magic Square**) Alice and Bob claim they can fill a 3×3 grid with $+1$ ’s and -1 ’s such that every row has a positive product while every column has a negative product. To verify this claim, the referee asks Alice to provide the entries of a randomly-chosen row of the grid and asks Bob to provide the entries of a randomly-chosen column. Alice and Bob “win” if and only if:

- ▶ The row has a positive product.
- ▶ The column has a negative product.
- ▶ The row and column agree on the square where they intersect.

As with the Odd Cycle game, there is no classical strategy that allows Alice and Bob to win with certainty, as that would require an actual assignment of entries on the grid that satisfies all the constraints on the rows and columns. The constraints on the rows require the total number of -1 ’s in the grid to be even, while the constraints on the columns require the number to be odd. This implies that there’s no classical strategy that lets Alice and Bob win the game with probability 1.

Nevertheless, David Mermin discovered a quantum strategy where Alice and Bob win with probability 1. This strategy requires them to share 2 ebits

of entanglement. We won't describe the strategy in detail in this class, since it is complicated to state without the additional concepts that we haven't yet seen. But, the Magic Square Game is just the tip of the iceberg of a very rich and interesting body of research on so-called **Quantum Psuedo-Telepathy Games**—that is, games with a perfect quantum strategy but no perfect classical strategy.

Lecture 15: Einstein-Certified Randomness

Until recently, the Bell inequality was taught because it was historically and conceptually important, not because it had any practical applications. Sure, it establishes that you can't get away with a local hidden variable theory, but in real life, no one *actually* wants to play the CHSH game, do they? Recently, however, the Bell inequality has found applications in one of the most important tasks in computing and cryptography: the generation of guaranteed random numbers.

15.1 Guaranteed Random Numbers

Generating random numbers is one of the most important tasks in computing (and certainly in cryptography). Once we have quantum mechanics you might think that the solution is trivial. After all, you can get a random bit by measuring the $|+\rangle$ state in the $\{|0\rangle, |1\rangle\}$ basis. Easy, right? But, this solution often isn't good enough for cryptography. Cryptographers are paranoid people and they want the ability to maintain security even if the hardware they're using was designed by their worst enemy.

These sorts of assumptions aren't just academic speculation, especially given the Snowden revelations. For example, NIST (the National Institute of Standards and Technology) put out a standard for pseudo-random number generation based on elliptic curves to be used for encryption. This standard was later discovered to have a backdoor, apparently created by the NSA, that would allow an attacker to predict the output numbers, thus being able to break systems encrypted under this standard.

Cryptographers want to base their random number generation on the smallest set of assumptions possible. They want bits that are guaranteed to be random, and to be sure that no one added predictability through any sort of backdoor. You might think that, logically, one can never prove that numbers are truly random: that the best one can ever say is “I can’t find any pattern here.” After all, you can’t prove a negative, and if not the NSA, who’s to say that the universe itself isn’t playing tricks on us by inserting a pseudorandom pattern into the workings of quantum mechanics?

Though presumably, if a higher power wanted to read our emails they could also do it some other way...

That’s what makes so interesting (and non-obvious) that the Bell inequality lets us certify numbers as being truly random under very weak assumptions. These assumptions basically boil down to “no faster-than-light travel is possible.” Let’s now explain how.

Suppose you have two boxes that share quantum entanglement. We’ll imagine the boxes were designed by your worst enemy, so you trust nothing about them. All we’ll assume is that the boxes can’t send signals back and forth, say because you put them in Faraday cages, or separated them by so large a distance that light doesn’t have enough time to travel between them during the duration of your interaction. A referee sends the boxes challenge numbers, x and y and the boxes return numbers a and b . If the returned numbers pass a test, we’ll declare them to be truly random.

So what’s the trick? Well, we already saw the trick; it’s just the CHSH game! The usual way to present the CHSH game is as a way for Alice and Bob to prove that they share entanglement, and thus that the universe is quantum-mechanical and that local hidden-variable theories are false. However, winning the CHSH game more than 75% of the time also establishes that a and b must have some randomness, and that there was some amount of entropy generated. *Why?* Because suppose instead that a and b were deterministic functions. That is, suppose they could be written as $a(x, r)$ and $b(y, r)$ respectively, in terms of Alice and Bob’s inputs as well as shared random bits. In that case, whatever these functions were, they’d define a local hidden-variable theory, which is precisely what Bell’s Theorem rules out! So the conclusion is that, if x and y are random and there’s no communication between Alice and Bob, then there must exist at least some randomness in the outputs a and b .

Around 2012, Umesh Vazirani coined the term **Einstein-Certified Randomness** for this sort of thing. The basic idea goes back earlier, for example, to Roger Colbeck’s 2006 PhD thesis and (in cruder form) to Prof. Aaronson’s

2002 review of Stephen Wolfram’s “A New Kind of Science,” which used the idea to refute Wolfram’s proposal for a deterministic hidden-variable theory underlying quantum mechanics.

OK, so how do we actually extract random bits from the results of the CHSH game? You could just take the stream of all the a ’s and b ’s that are outputted after many plays of the CHSH game. Admittedly, this need not give us a *uniform* random string. In other words, if the output string has length n , then its Shannon entropy,

$$\sum_x p_x \log_2 \frac{1}{p_x} \quad (15.1)$$

where p_x is the probability of string x , will in general be less than n . However, we can then convert x into an (almost) uniformly random string on a smaller number of bits, say $\frac{n}{10}$ or something, by using a well-known tool from classical theoretical computer science called a *randomness extractor*. A randomness extractor is a function that crunches down many sort-of-random bits (and, typically, a tiny number of truly random bits, called the *seed*) into a smaller number of very random bits.

If you’re interested in learning more about these, consider taking a course with David Zuckerman (here at UT) who is an expert on randomness extractors.

OK, but there’s an obvious problem with this whole scheme. Namely, we needed the input bits to be uniformly random in order to play the CHSH game. But, that means we put in two perfect random bits, x and y , in order to get out two bits a and b that are not perfectly random! In other words, the entropy we put in is greater than the entropy we get out, and the whole thing is a net loss. A paper by Pironio et al. addressed this by pointing out that you don’t have to give Alice and Bob perfectly random bits every time the CHSH game is played. Instead, you can just input $x = y = 0$ most of the time, and occasionally stick in some random x ’s and y ’s to prevent Alice and Bob from using hidden variables. Crucially, if Alice or Bob gets a 0 input in a given round, then they have no way of knowing whether that round is for testing or for randomness generation. So, if they want to pass the randomly-inserted tests, then they’ll need to play the CHSH game correctly in *all* the rounds (or almost all of them), which means generating a lot of randomness.

At this point it all comes down to a quantitative question of *how much entropy can we get out, per bit of entropy that we put in?* There was a race to answer this by designing better and better protocols that got more and

more randomness out per bit of randomness invested. First, Colbeck showed how to get cn bits out for n bits in, for some constant $c > 1$. Then, Pironio et al. showed how to get $\sim n^2$ bits out per n bits in. Then, Vazirani and Vidick showed how to get $\sim e^{\sqrt{n}}$ bits out per n bits in, which is the first time we had exponential randomness expansion. But, all this time an obvious question remained in the background: *why not just use a constant amount of randomness to jump-start the randomness generation, and then feed the randomness outputted by Alice and Bob back in as input, and so on forever, thereby getting unlimited randomness out?*

It turns out that a naïve way of doing this doesn't work. If you just feed Alice and Bob the same random bits that they themselves generated, then they'll *recognize* those bits, so they won't be random *to them*. This will allow Alice and Bob to cheat, making their further outputs non-random.

Remember: We're working under the assumption that "Alice" and "Bob" are machines designed by our worst enemy!

If you don't have a limit on the number of devices used, then a simple fix for this problem is to feed Alice and Bob's outputs to two other machines, Charlie and Diane. Then you can feed Charlie and Diane's outputs to two more machines, Edith and Fay, and so on forever, getting exponentially more randomness each time. But what if we have only a *fixed* number of devices (like 4, or 6) and we still want unlimited randomness expansion? In that case, a few years ago Coudron and Yuen, and independently Chung, Miller, Shi, and Wu, figured out how to use the additional devices as "randomness laundering machines." These extra machines are used to convert random bits that Alice and Bob can predict into random bits that they can't predict, so that the output bits can be fed back to Alice and Bob for further expansion.

One question that these breakthrough works *didn't* address was exactly how many random seed bits are needed to jump-start this entire process. Like, are we talking a billion bits or 2 bits? In a student project supervised by Prof. Aaronson, Renan Gross calculated the first explicit upper bound, showing that a few tens of thousands of random bits suffice. That's likely still far from the truth, and finding a tighter upper-bound is still an open question. For all we know it might be possible with as few as 10 or 20 random bits.

It's also worth mentioning that this sort of protocol has already been experimentally demonstrated at NIST, and indeed is part of what's used for NIST's public randomness beacon.

15.1.1 Leashing Quantum Systems

You might wonder *what else can you certify about two separated boxes, by seeing them win at the CHSH game?* It turns out that the answer is an *enormous* number of things. In a tour-de-force paper from 2012, Reichardt, Unger, and Vazirani showed how to use a sequence of CHSH-like challenges to certify that Alice and Bob performed a specific sequence of unitary transformations on their qubits (up to local changes of basis). This means that, just by making Alice and Bob repeatedly win the CHSH game, you can *force them to do any quantum computation of your choice*. Reichardt et al. describe this as a “classical leash for a quantum system.”

This sort of thing constitutes one of the main current ideas for how a classical skeptic could verify the operation of a quantum computer. For a problem like factoring a huge integer into primes, for example, we can easily verify the output of a quantum algorithm by simply multiplying the claimed factors and seeing if they work! But this isn’t believed to be the case for all problems that a quantum computer can efficiently solve. Sometimes the best way to efficiently verify a quantum computer is working correctly might involve using quantum resources yourself. What Reichardt et al. show is that, as long as we have *two* quantum computers that are entangled with each other, but unable to exchange messages, we can use the CHSH game to verify that the computers are behaving as expected.

It turns out, however, that even if we only have a single quantum computer and a classical verifier, it is still possible to leash the quantum computer and force it to perform any computation we want. In a groundbreaking paper from 2018, Urmila Mahadev (then a graduate student at UC Berkeley) developed such a protocol. She showed that so long as we are willing to accept some standard cryptographic, we could use cryptographic techniques to force a quantum computer to perform any desired computation., or else catch it cheating.

This brings us nicely to quantum computation, which is probably the subject that most of you took the course to learn about! We’ll begin discussing quantum computation in earnest in the next lecture.

Lecture 16: Quantum Computing and Universal Gate Sets

Having seen lots of quantum protocols, we're finally ready to tackle the holy grail of the field: a programmable quantum computer, a single machine that could do any series of quantum-mechanical operations. Quantum computation has two intellectual origins. One comes from David Deutsch, who was thinking about experimentally testing the Many-Worlds Interpretation (of which he was and remains a firm believer) during his time as a postdoc here at UT Austin¹. Much like with Wigner's friend, Deutsch imagined creating an equal superposition of a brain having measured a qubit as $|0\rangle$, and the same brain having measured the qubit as $|1\rangle$. In some sense, if you ever had to ascribe such a superposition state to yourself then you'd have gone beyond the Copenhagen Interpretation. *But, how could we ever test this?* Given the practical impossibility of isolating all the degrees of freedom in a human brain, the first step would presumably have to be: take a complete description of a human brain, and upload it to a computer. Then, the second step would be to put the computer into a superposition of states corresponding to different thoughts.

This then naturally leads to more general questions: *could anything as complicated as a computer be maintained in a superposition of "semantically different" states? How would you arrange that, in practice? Would such a computer be able to use its superposition power to do anything that it couldn't do otherwise?*

The other path to the same questions came from Richard Feynman, who gave a famous lecture in 1982 concerned with the question *how do you simulate quantum mechanics on a classical computer?* Chemists and physicists had known for decades that this is hard, essentially because the number of amplitudes that you need to keep track of increases exponentially with the number of particles. This is the case because, as we know, an n -qubit state in

¹David Deutsch is arguably more famous, however, for his pivotal role in the Avengers' defeat of the evil tyrant Thanos, in the movie *Avengers Endgame*.

the most general case requires 2^n amplitudes to specify completely. Chemists and physicists know many approximation methods for such simulation problems. These include methods such as Density Functional Theory and Quantum Monte Carlo, which often work well in practice. In the worst case though, there's no known shortcut to dealing with the whole vector of amplitudes. So Feynman raised the question, *Why don't we build computers out of qubits, which themselves could take advantage of superposition and interference and entanglement?* Of course he then faced the question: *supposing we built such a computer, what would it be useful for?* At that time, he was really only able to suggest one answer to that question. Namely, it would be good for simulating quantum mechanics itself! More specifically, he ended his talk with the following:

Nature isn't classical, dammit, and if you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy

If and when quantum computers really become practical quantum simulation is arguably still the most important application we know for them. In any case, no one knew at the time whether quantum computers would be useful for “classical” tasks as well; that's a different, harder question, which will occupy us for much of the rest of the course.

We already have all the tools we need to discuss quantum computers. The basic picture of a quantum computer is that it's just a quantum circuit, but we're jumping from working with 1, 2, or 3 qubits at a time to n qubits—where n could be, say, a million or more. You apply a sequence of gates on these qubits, each gate acting on just a few qubits (say, 1 or 2 of them), then measure some or all of the qubits to get your result.

Let's address a few conceptual questions before proceeding further:

Will we be able to solve any problem on a quantum computer that literally can't be solved on a classical computer, regardless of resources? It's easy to see that the answer is no. Using a classical computer one can always simulate a quantum computer with n qubits by just explicitly storing the vector of 2^n amplitudes (to some suitable precision, enough to approximate the final probabilities), and updating the vector whenever a unitary transformation gets applied. For this reason, we see that a quantum computer could “only, at most” achieve an exponential speedup over a classical computer; it could *only* falsify the Extended Church-Turing Thesis, rather than the original Church-Turing Thesis. Quantum computers might change what's *computable in polynomial time*—how drastically, we don't yet know—but they're not going to change

what's computable in general. As such, they certainly can't let us solve the halting problem or anything of that kind.

Why does each gate act on only a few qubits? Where is this assumption coming from?

It's similar to how classical computers don't have gates act on arbitrarily large numbers of bits, and instead use small gates like AND, OR, and NOT to build up complex circuits. The laws of physics provide us with *local* interactions—one particle colliding with another one, two currents flowing into a transistor, etc. . . —and it's up to us to string together those local interactions into something more complicated.

In the quantum case, you could imagine a giant unitary matrix U , which takes qubits, interprets them as encoding an instance of the 3SAT problem (or some other **NP**-complete problem) and then CNOTs the answer (1 for yes, 0 for no) into an extra qubit. That U formally exists and is formally allowed by the rules of quantum mechanics. *But how would you go about actually implementing it in practice?* Well, you'd need to build it up out of smaller components—say, components that act on only a few qubits at a time.

It turns out that it's possible to implement any unitary U you want using exponentially many simple components (we'll say more about that later in the lecture). The question that will most interest us in quantum computing theory is which U 's can be implemented using only *polynomially* many simple components. Those are the U 's that we'll consider *feasible* or *efficient*.

What is the role of interference in quantum computing?

Quantum amplitudes can cancel each other out; that's the most important way in which they differ from classical probabilities. The goal in quantum computing is always to choreograph a pattern of interference such that, for each wrong answer, some of the contributions to its amplitude are positive and others are negative (or, for complex amplitudes, they point every which way in the complex plane), so on the whole they *interfere destructively* and cancel each other out. Meanwhile, the contributions to the correct answer's amplitude should *interfere constructively* (say, by pointing along the same direction in the complex plane). If we can arrange that, then when we measure, the right answer will be observed with high probability.

Note that if it weren't for interference, then we might as well have just used a classical computer with a random-number generator and saved the effort of building a quantum computer. In that sense, all quantum-computational advantage relies on interference.

What is the role of entanglement in quantum computing?

In some sense, we can develop the entire theory of quantum computing without ever talking about entanglement. However, pretty much any time

we’re doing an interesting quantum computation, entanglement is something that will be there. The reason for this is simply that an unentangled pure state of n qubits can always be written as

$$(\alpha_0 |0\rangle + \beta_0 |1\rangle) \otimes (\alpha_1 |0\rangle + \beta_1 |1\rangle) \otimes \cdots \otimes (\alpha_{n-1} |0\rangle + \beta_{n-1} |1\rangle)$$

Specifying such a state requires only $2n$ amplitudes, so we can store it efficiently. Thus, if for some reason the (pure) state of our quantum computer never had any entanglement in it, then a classical computer would be able to simulate it efficiently, and would be unable to achieve any speedup. By contrast, a general entangled state of n qubits,

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x, \quad (16.1)$$

requires 2^n amplitudes to specify. It very quickly becomes hopelessly intractable to store and manipulate all these amplitudes on a classical computer.

With 300 qubits, we already have more amplitudes to deal with than there are atoms in the observable universe.

16.1 Complexity of General Unitaries: Counting Argument

It’s a theorem, which we won’t prove in this class, that any unitary transformation on *any* number of qubits can be decomposed as a product of 1- and 2-qubit gates. However, if you just run the decomposition blindly, it will produce a quantum circuit with something like 4^n gates—just like, if you use the method of truth tables to build a circuit to compute some arbitrary Boolean function, $f : \{0, 1\}^n \rightarrow \{0, 1\}$, you’ll get something with about 2^n AND, OR, and NOT gates. Just like in the classical world, our real interest is in which Boolean functions that can be *efficiently* computed—say, using a polynomial number of AND, OR, and NOT gates—so too in the quantum world, our real interest is in which n -qubit unitary transformations can be realized using only polynomially many 1 and 2-qubit gates.

That being so, it behooves us to ask: *is it possible that all n -qubit unitaries can be realized by polynomial-size circuits?* The answer turns out to be no. In fact, just like “almost all” Boolean functions require exponentially large

circuits to compute them, so too “almost all” unitaries require exponentially large quantum circuits. As in the classical case, the way to prove this is using a counting argument.

Counting arguments in circuit complexity go back to Claude Shannon in 1949. Shannon observed that almost every n -bit Boolean function requires a circuit of at least $\sim \frac{2^n}{n}$ AND, OR and NOT (or equivalently NAND) gates to compute it. The reason for this, in one sentence, is that there are too many Boolean functions, and not enough small circuits! In more detail, there are 2^{2^n} different Boolean functions $f : \{0, 1\}^n \rightarrow \{0, 1\}$. For a circuit with T gates there are $\sim (n + T)^{\mathcal{O}(T)}$ different circuits that can be constructed, which can be seen as follows. Without loss of generality suppose we choose to construct our circuit out of NAND gates, with a bounded fan-in of 2. For each NAND gate in our circuit we have

$$\binom{N + T - 1}{2} \leq (N + T)^2$$

ways to select the inputs. Since there are T gates, the number of different circuits we can construct this way is upper bounded by

$$(N + T)^{2T} = (N + T)^{\mathcal{O}(T)}. \quad (16.2)$$

Since each circuit can only compute one function, we can simply set the two expressions equal and solve for T .

$$2^{2^n} = (n + T)^T \approx T^T \quad (16.3)$$

Taking the log of both sides gives

$$2^n = T \log T \quad (16.4)$$

This is satisfied for $T \approx \frac{2^n}{n}$. Moreover, if T is smaller then $(N + T)^{\mathcal{O}(T)}$ is minuscule compared to 2^{2^n} , so almost every function must require a circuit of size $\mathcal{O}(\frac{2^n}{n})$.

Strikingly, and famously, this argument doesn't give us a single example of a hard-to-compute Boolean function. It merely tells us that such functions must exist, and indeed are ubiquitous!

We can use a similar sort of argument in the quantum case—although, since $2^n \times 2^n$ unitary matrices form a continuous manifold, it's easier to talk in terms of dimensionality rather than cardinality. For simplicity, let's even

restrict attention to those $2^n \times 2^n$ unitary matrices that are *diagonal*. Even then, specifying such a matrix clearly requires us to specify 2^n independent complex numbers of norm 1 (or $2^n - 1$, if we ignore global phase). By contrast, a quantum circuit with T gates, each acting on at most 2 qubits, can be specified using only $\mathcal{O}(T)$ continuous parameters (plus some discrete choices about where to apply the gates, which won't affect the argument). So we have a 2^n -dimensional manifold in the one case, and a union of $\mathcal{O}(T)$ -dimensional manifolds in the other. Clearly, for the one to cover the other, we need T to grow at least like $\sim 2^n$. Hence there exist n -qubit unitary transformations that require exponentially many gates to implement. In fact, “almost all” (now in the technical, measure-theory sense of 100% of them) have this property.

You might complain that we've only showed that exponentially many gates are needed to implement most n -qubit unitary transformations exactly. *What about approximately implementing them? That is, implementing them up to some small error ϵ (say, in each entry)?* In fact, a little algebraic geometry (which we won't go into here) is enough to show that exponentially many gates are needed to approximate most n -qubit unitaries as well, or even most n -qubit diagonal unitary transformations.

Once again, these arguments don't give us a single example of a hard-to-implement unitary transformation. They just tell us that that this is the norm. The easy-to-implement unitaries are the rare exceptions! Yet, rare though they might be, the subset of unitaries that are easy (i.e., that can be implemented by polynomial-size quantum circuits) are the main ones that will interest us in quantum computing.

16.2 Universal Gate Sets

Up till now, we've assumed that any 1 and 2-qubit gates are available, but it turns out that assumption isn't actually necessary. This brings us to our next topic, **Universal Gate Sets** for quantum computing.

16.2.1 Classical Universality

In classical computing, you're probably familiar with logic gates like AND, OR, NOT, NAND, etc. A (classical) gate set is called *universal* if, by stringing together enough gates from the set, you can express any Boolean function on

any number of bits. For example, the NAND gate by itself is universal. The diagram in Figure 16.1 shows how you'd construct an OR gate out of NANDs. You can also work out how to construct an AND gate and a NOT gate, and from there you can get anything else. By contrast, the set $\{\text{AND}, \text{OR}\}$ is not universal, because it can only express *monotone* Boolean functions; that is, changing an input bit from 0 to 1 can never change an output bit from 1 to 0. Likewise, the set $\{\text{NOT}, \text{XOR}\}$ is not universal, because it can only express Boolean functions that are linear or affine mod 2. So, while “most” gate sets are universal, being universal isn't completely automatic.

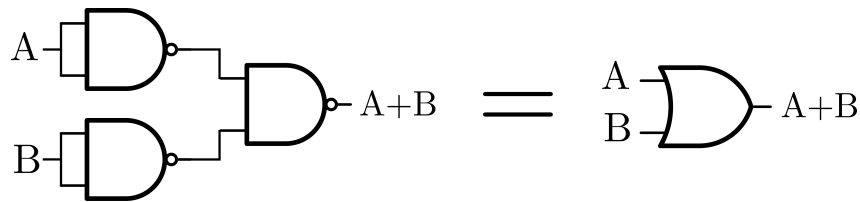


Figure 16.1: Simulating an OR gate using NAND gates.

Next let's discuss *classical reversible gates*, reversible mappings from n -bit strings to n -bit strings. We call a set of reversible gates universal if, by composing enough of them, you can express any reversible transformation on any number of bits. Here we allow the use of “ancilla bits” (extra bits used as a resource to implement a particular gate), as long as the ancilla bits are returned to their initial states by the end of the computation. The use of ancilla bits is provably necessary for universality in this setting.

The most famous example of a universal reversible gate—the reversible analogue of the NAND gate—is called the **Toffoli Gate**. The Toffoli gate, also known as controlled-controlled-NOT, is a 3-bit gate that flips the third bit iff the first two bits are both set to 1.

To show that Toffoli is capable of universal computation, we construct a NAND gate out of a Toffoli gate in the diagram in Figure 16.2. Because Toffoli can simulate NAND, it can also simulate any ordinary (non-reversible) Boolean circuit (given a sufficient number of ancillary bits).

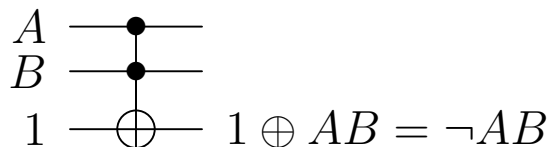


Figure 16.2: Circuit for simulating the effect of NAND using a Toffoli gate along with an ancillary bit.

Note that this argument does not yet establish that Toffoli is a universal reversible gate in the sense we defined above, because for that we need to implement all possible reversible transformations, not just compute all Boolean functions. However, a more careful argument, part of which we'll give later in the course, shows that Toffoli is universal in that stronger sense as well.

A good example of a gate that separates the two kinds of universality is the **Fredkin Gate**, which is also called Controlled-SWAP or CSWAP. This is a 3-bit gate that swaps the second and third bits iff the first bit is set to 1. Just like the Toffoli gate, the Fredkin gate can simulate a NAND gate (we'll leave the construction as an exercise for the reader) and is therefore capable of universal computation. However, the Fredkin gate is not universal in the stronger sense, because it can never change the total number of 1's in the input string, also called the *Hamming weight*. For example, it's impossible to compose any number of Fredkin gates in order to map the string 000 to 111. A gate with the property that it always preserves the Hamming weight of the input is called *conservative*.

There are also reversible gates that are not even capable, on their own, of universal computation. An important example is the CNOT gate, which we saw earlier. By composing CNOT gates, we can only express Boolean functions that are affine mod 2. For example, we can never express AND or OR. More generally, and in contrast to the irreversible case, it turns out that *no* 2-bit classical reversible gate is capable of universal computation. For universality, we need reversible gates (such as Toffoli or Fredkin) that act on at least 3 bits.

It's worth noting that any classical reversible gate can also be used as a quantum gate (i.e., it's unitary). From this we can immediately see that, if nothing else, a quantum circuit can compute any function that a classical circuit of similar size can compute. We simply need to transform the classical circuit into one made of, say, Toffoli gates.

16.2.2 Quantum Universality

We'll call a set of quantum gates \mathcal{S} *universal* if, by composing gates from \mathcal{S} , you can approximate any unitary transformation on any number of qubits to any desired precision. Note that if \mathcal{S} is finite then approximation is all we can hope for, because there are uncountably many unitary transformations, but only a countable infinity of quantum circuits that we can build using finitely many gates from \mathcal{S} . Just like with classical reversible gates, there are weaker kinds of universality for quantum gate sets that are often enough in practice. That is, even if gates *can't* be used to approximate an arbitrary unitary to any desired precision, they'll often suffice anyway for universal quantum

computation. We'll see examples of this soon.

Let's list some of the ways a quantum gate set could be limited, in such a way that it *fails* to be universal. We'll discuss four such ways, three of them obvious and one of them not.

- ▶ Your gate set doesn't create interference/superposition.
 - Example: The set $\mathcal{S} = \{\text{CNOT}\}$ can only map computational basis states, like $|10\rangle$, to other computational basis states, like $|11\rangle$. It can maintain existing superpositions, but it can't *create* a superposition of basis states where there wasn't one already.
- ▶ Your gate set can create superpositions, but not entanglement.
 - Example: The set $\mathcal{S} = \{\text{Hadamard}\}$ can map $|0\rangle$ to $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$, thereby creating a superposition. But, it should be obvious that the Hadamard gate can't map a product state to an entangled state, since it acts on only one qubit. In fact, any quantum circuit made of only Hadamard gates—or any 1-qubit gates, for that matter—will just act independently on each of the n qubits, so it will be trivial to simulate classically.
- ▶ Your gate set only has real gates.
 - Example: The set $\mathcal{S} = \{\text{CNOT}, \text{Hadamard}\}$ is getting closer to universality, as it's capable of creating entangled superposition states. But, the CNOT and Hadamard matrices have real entries only, so composing them could never give us a unitary transformation like

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

- ▶ Your gate set is “contained in the stabilizer set.”
 - This is the non-obvious case. Later in the course, we'll explain stabilizer gates in more detail, as well as their central importance for quantum error correction. For now, though, the stabilizer gates are the following three quantum gates: CNOT, Hadamard, and S , where S is the matrix above. These gates pass every test for universality that we saw before: they can generate superposition, and entanglement, and complex amplitudes. Furthermore, they're enough to demonstrate many of the quantum phenomena that we've seen in this course, such as teleportation and superdense coding. Nevertheless, it turns out that the particular set

$$\mathcal{S} = \{\text{CNOT}, \text{Hadamard}, S\}$$

is not universal. Indeed, a famous result called the **Gottesman-Knill Theorem** shows that these gates generate only a discrete subset of unitary transformations and that furthermore, any circuit consisting only of stabilizer gates, applied to the initial state $|0 \cdots 0\rangle$, can be simulated in polynomial time by a classical computer. Thus, despite containing an interesting subset of quantum mechanics, these gates still aren't enough to realize exponential quantum speedups.

Are there any other ways for a set of quantum gates, acting on qubits, to fail to be universal? That's currently an open question! It's one of Prof. Aaronson's personal favorites. Not many people in the field care about this particular question, since we have lots of universal gate sets that work, and so most just roll with it. But it would be nice to know, and you should go solve it.

So then, what are some examples of universal quantum gate sets? It turns out that the stabilizer set, $\mathcal{S} = \{\text{CNOT}, \text{Hadamard}, S\}$, becomes universal if you swap out the Hadamard gate for nearly anything else. So, for example, $\mathcal{S} = \{\text{CNOT}, R_{\pi/8}, S\}$ is universal, as is $\mathcal{S} = \{\text{Toffoli}, \text{Hadamard}, S\}$ by a 2002 result of Yaoyun Shi. Additionally, if you just pick a 2-qubit gate uniformly at random, then it's known to have a 100% chance (in the measure-theoretic sense) of being universal. With universality, the whole difficulty comes from the remaining 0% of gates! Above, we listed bad cases—ways of failing to be universal—but there are also general criteria such that if your gate set meets them then it's universal. We won't cover this, but see the paper of Shi, for example, for such criteria.

16.2.3 The Solovay-Kitaev Theorem

So far, in discussing universal gate sets, we've swept an important question under the rug. Namely, a universal gate set lets us approximate any unitary to any desired accuracy ϵ , but *how does the number of gates scale with respect to ϵ ?* If, for example, the number scaled like $2^{\frac{1}{\epsilon}}$, then our gate set would be “universal,” but not especially *useful*. Fortunately, there's a central result in quantum computing theory called the **Solovay-Kitaev Theorem**, that shows the number of gates needed to approximate a unitary to precision ϵ scales manageably.

The Solovay-Kitaev Theorem says that, using any universal gate set \mathcal{S} that's closed under inverses (that is, if $G \in \mathcal{S}$ then so is G^{-1}), we can approximate any unitary on qubits to within precision ϵ (say, entrywise precision)

using only $\mathcal{O}(4^n \text{polylog}(\frac{1}{\epsilon}))$ gates. In other words, if we treat n as fixed, then the complexity scales only like some power of $\log(\frac{1}{\epsilon})$. This means that *all* universal gate sets, or at least the ones closed under inverses, “fill in the space of all unitary transformations reasonably quickly.” Furthermore, the gate sequences that achieve the Solovay-Kitaev bound can actually be found by a reasonably fast algorithm. Whether the closed-under-inverses condition can be removed remains an unsolved problem to this day.

The original proofs of Solovay-Kitaev from the late 1990s required a number of gates which grew like $\log^{3.97}(\frac{1}{\epsilon})$. However, more recently it's been shown that, at least if we use special universal gate sets arising from algebra and number theory, we can get the number of gates to grow only like $\log(\frac{1}{\epsilon})$. This is not only much more practical, but it's also the best one could possibly hope for on information-theoretic grounds. The proof of the Solovay-Kitaev Theorem is beyond the scope of this course, but it's contained in many quantum computing textbooks, including Nielsen & Chuang.

Lecture 17: Quantum Query Complexity and The Deutsch-Josza Problem

People often want to know where the true power of quantum computing comes from.

- ▶ *Is it the ability of amplitudes to interfere with one another?*
- ▶ *Is it the huge size of Hilbert space (the space of possible quantum states)?*
- ▶ *Is it that entanglement gives us 2^n amplitudes to work with?*

But that's sort of like dropping your keys and asking *what made them fall?*

- ▶ *Is it their proximity to the Earth?*
- ▶ *Is it the curvature of spacetime?*
- ▶ *Is it the fact that you dropped them?*

There can be many complementary explanations for the same fact, all of them valid, and that's the case here. If there weren't a huge number of amplitudes, quantum mechanics would be easy to simulate classically. If the amplitudes were instead standard probabilities, rather than complex numbers that could interfere with each other, QM would also be easy to simulate classically. If no entanglement were allowed, then all pure states would be product states, once again easy to represent and simulate classically. If we were restricted to stabilizer operations, QM would be easy to simulate classically by the Gottesman-Knill Theorem. But as far as we know, full QM—involving interference among exponentially many amplitudes in complicated, entangled states and with non-stabilizer group operations—is hard to simulate classically, and that's what opens up the possibility of getting exponential speedups using a quantum computer.

17.1 Quantum Query Complexity

There are two major ways we look at the complexity of quantum algorithms. The **Circuit Complexity** of a unitary transformation, U , is the size (i.e., number of gates) of the smallest circuit that implements U . We like unitaries with polynomial circuit complexity. Alas, typically it's *extremely* hard to determine the circuit complexity of a unitary; the best we can do is to prove upper bounds, and conjecture lower bounds on the basis of hardness assumptions and reduction arguments. Note that the reasons why this sort of problem is insanely hard have nothing to do with quantum mechanics.

“What’s the smallest circuit that solves Boolean satisfiability?” is a similarly hard problem, indeed closely related to P vs NP .

Given the difficulty of determining the circuit complexity, we often make use of an alternative, albeit more limited model, of complexity. This alternative model is called **Query Complexity**. Here we count the number of calls an algorithm makes to an oracle (or black box function). The idea is that your oracle takes an input x and produces an output $f(x)$, where $f : \{0, 1\}^n \rightarrow \{0, 1\}$. In quantum mechanics, our first guess for what this would mean might be that we map the input state $|x\rangle$ to the output state $|f(x)\rangle$, or maybe the output state $|x\rangle|f(x)\rangle$.

Here, though, we run into trouble because such a transformation is not unitary. To make the transformation unitary we need a so-called *answer* or *target* register. So we give the black box two inputs, x , which is unchanged, and y , which has the answer $f(x)$ written into it in a reversible way:

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

If we took care to ensure that $y = 0$ initially, then this would map $|x, 0\rangle$ to $|x, f(x)\rangle$, exactly like in the classical case. We'll call an oracle constructed in this way a **XOR Oracle**.

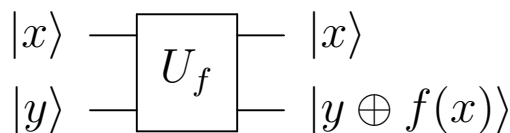


Figure 17.1: Diagram of the XOR oracle.

Later, we'll see that it's often most convenient to consider quantum queries

that map each basis state $|x\rangle$ to $(-1)^{f(x)}|x\rangle$. In other words, queries that write the function value into the phase of the amplitude, rather than storing it explicitly in memory. Or, more precisely, we consider queries that map each basis state $|x, b\rangle$ to $(-1)^{f(x)\cdot b}|x, b\rangle$, where b is a bit that controls whether the query should even take place or not. This sort of **Phase Oracle** doesn't really have any classical counterpart, but it is extremely useful for setting up desired interference patterns.

It behooves us to ask: *how do the phase oracle and XOR oracle compare? Could one be more powerful than the other?* Happily, it turns out that they're equivalent, in the sense that either can be used to simulate the other with no cost in the number of queries. This is a result that we'll need later in this lecture. To see the equivalence, all we need to do is consider what happens when the second register is placed in the $|-\rangle$ state before a query using a Hadamard gate. You can check that this converts a XOR oracle into a phase oracle as follows:

$$\frac{|x, 0\rangle - |x, 1\rangle}{\sqrt{2}} \rightarrow \frac{|x, 0 \oplus f(x)\rangle - |x, 1 \oplus f(x)\rangle}{\sqrt{2}} \quad (17.1)$$

if $f(x) = 0$ then the state is

$$\frac{|x, 0\rangle - |x, 1\rangle}{\sqrt{2}} = |x, -\rangle \quad (17.2)$$

and if $f(x) = 1$ then the state is

$$\frac{|x, 1\rangle - |x, 0\rangle}{\sqrt{2}} = -|x, -\rangle \quad (17.3)$$

we can rewrite both outcomes as just $(-1)^{f(x)}|x, -\rangle$. Meanwhile, if the second register is placed in the $|+\rangle$ state, then nothing happens, so we really do get the $|x, b\rangle \rightarrow (-1)^{f(x)\cdot b}|x, b\rangle$ behavior. Conveniently, the converse is also true! That is, if we have a phase oracle, then by placing the output register in one of the states $|+\rangle$ or $|-\rangle$, we can simulate the effect of a XOR oracle, with the phase oracle causing $|+\rangle$ and $|-\rangle$ to be swapped if and only if $f(x) = 1$.

Taking a step back, though, what are we really doing when we design a quantum algorithm in the query model? We're abstracting away part of the problem, by saying "Our problem involves some function, $f(x)$, that we want to learn some property of in a way that only requires evaluating $f(x)$ on various inputs, while ignoring the details of how f is computed." Because of this abstracting away of details, the query model is also referred to as the **Black-Box Model**. So, for example, you might want to learn: *is there some*

input x such that $f(x) = 1$? Does $f(x) = 1$ for the majority of x 's? Does f satisfy some global symmetry, such as periodicity, or is it far from any function satisfying the symmetry? etc. . . We then want to know how many queries to f are needed to learn this property.

In this model we ignore the cost of any quantum gates that are needed before or after the queries; those are treated as "free."

Why do we care about the black-box model? You're debating how you'd phrase your wishes if you found a magical genie. Who cares? The truth is more prosaic, though. You can think of a black box as basically a huge input string. From that standpoint, querying $f(x)$ just means looking up the x^{th} element in the string. Or another perspective is to imagine you're writing code that calls a subroutine that computes $f(x)$. You don't want to modify the subroutine (or even examine it's internal workings). You just want to know how many calls to it are needed to find some information about f . We assume in the quantum case that calls to the subroutine can happen on a superposition of different input values and that we get back a superposition of different answers.

17.2 Quantum Garbage Collection

To justify the quantum black-box model there's one technical question we need to answer. Suppose we did have a small circuit to compute a function f . *Could we then implement the quantum black-box behavior that we described above? That is, could we implement the behavior of an XOR oracle?* The reason this isn't entirely obvious is that quantum circuits have to be *reversible*. So just because there's a small circuit to compute f , doesn't immediately imply that there's a small circuit that maps the basis state $|x, y\rangle$ to the basis state $|x, y \oplus f(x)\rangle$ while leaving nothing else lying around.

Indeed, let's step back, and think about the constraints on computation that are imposed by reversibility. To start with the obvious, if we had a reversible circuit that maps $|x\rangle$ to $|f(x)\rangle$, then f must be an *injective* function. But, now for the subtle part, even if $f(x)$ is both injective *and* efficiently computable, that still doesn't imply (at least, as far as we know) that the map $|x\rangle \rightarrow |f(x)\rangle$ is efficiently computable. *Why not?* Well, imagine that $f(x)$ is an *injective one-way function*, a function that's easy to compute but hard to invert. Such functions are the basic building blocks of cryptography and are

strongly conjectured to exist, even in a world with quantum computers.

Note that even though quantum computers can break a few supposedly one-way functions, like those based on factoring and discrete log, there are many, many more “generic, less structured” one-way functions that don’t seem threatened by quantum computers. We’ll have more to say about such issues later.

Now suppose we had a small circuit \mathcal{C} such that $\mathcal{C}|x\rangle = |f(x)\rangle$. Then, simply by running that circuit backwards (inverting all the gates and reversing their order) we could get $\mathcal{C}^{-1}|f(x)\rangle = |x\rangle$. Thereby inverting the supposed one-way function! *But why doesn’t this contradict our starting assumption that f was easy to compute?* Because a reversible circuit for f would at best give us a mapping like $|x\rangle|0\rangle \rightarrow |x\rangle|f(x)\rangle$, a mapping that leaves x around afterward. Inverting that mapping will only take us from $f(x)$ to x if we *know x already*, so the reversible mapping is no help in breaking the one-way function after all.

This still leaves the question of how we efficiently implement the mapping $|x\rangle|0\rangle \rightarrow |x\rangle|f(x)\rangle$ if given a small non-reversible circuit for f . In the last lecture we saw how it’s possible to take any non-reversible circuit and simulate it by a reversible one by, for example, using Toffoli gates to simulate NAND gates. The trouble is that along the way to the final answer, the construction also produces all sorts of undesired results in the ancillary bits. The technical name for this is **Garbage** (yes, really).

Why is garbage such a problem for quantum computing? Because garbage can prevent the desired interference patterns from showing up, and the whole point of quantum algorithms is to create those interference patterns.

For example, what’s the difference between having $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and having $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$, where we treat the second qubit as unwanted garbage? The garbage is entangled with the first qubit, which is the qubit we care about. In the second case, when we look at only the first qubit, we see the maximally mixed state rather than the $|+\rangle$ state we wanted.

Returning to the original question, suppose you have a circuit to compute f . *How you we get a circuit that maps*

$$\sum_x \alpha_x |x, 0\rangle \rightarrow \sum_x \alpha_x |x, f(x)\rangle$$

without all the garbage? Back in the 1970s, Charles Bennett (while studying this problem in the context of classical reversible computation) invented a trick for this called **Uncomputing**. It's simple, though also strange when you first see it. The trick to getting rid of garbage is to run computations first forward and then in reverse. Let's say I have some circuit \mathcal{C} such that

$$\mathcal{C} |x, 0, \dots, 0\rangle \rightarrow |x, \text{gar}(x), f(x)\rangle,$$

where $\text{gar}(x)$ is a generic term for all the garbage produced as a byproduct of the computation. Then I do the following:

- ▶ First run the circuit \mathcal{C} to get $|x, \text{gar}(x), f(x)\rangle$.
- ▶ Then CNOT the answer $f(x)$ into a register initialized to 0 to get $|x, \text{gar}(x), f(x), f(x)\rangle$ (in other words, make a copy of $f(x)$ in a safe place).
- ▶ Finally, run the inverse circuit, \mathcal{C}^{-1} , to get $|x, 0, \dots, 0, f(x)\rangle$, or just $|x, f(x)\rangle$ if we ignore the 0's.

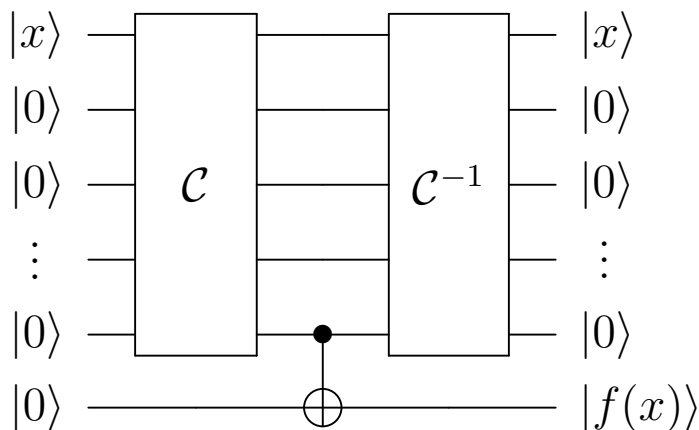


Figure 17.2: Circuit depicting the uncomputation of garbage.

The reason why we can copy $f(x)$ in spite of the No-Cloning Theorem is that we're assuming that $f(x)$ is a classical answer. This won't work if the output of the circuit is a general quantum state.

With uncomputing out of the way we're finally ready to talk about some quantum algorithms.

17.3 Deutsch's Algorithm

Deutsch's Algorithm was, by some definitions, the first quantum algorithm. It was proposed in the mid-1980s by David Deutsch and while its speedup is perhaps initially unimpressive, it makes use of many of the same building blocks that we'll reuse later on. Deutsch's algorithm computes the parity of two bits using only one (superposed) query to the bits.

In more detail, suppose we're given oracle access to two unknown bits, $f(0)$ and $f(1)$. Given an index $x \in \{0, 1\}$, our oracle returns the bit $f(x)$. Classically this would clearly take two queries since we need to know both bits. Using a quantum algorithm we can do it in one.

Let's start with a qubit initialized to $|0\rangle$. We'll then apply a Hadamard gate to it to get $|+\rangle$. Next we'll apply a phase query, which multiplies the amplitude of each basis state $|x\rangle$ by $(-1)^{f(x)}$. This yields:

$$|0\rangle \rightarrow \frac{|0\rangle + |1\rangle}{\sqrt{2}} \rightarrow \frac{(-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle}{\sqrt{2}} \quad (17.4)$$

So now, if $f(0) = f(1)$, that is we have even parity, we get

$$(-1)^{f(0)} \frac{|0\rangle + |1\rangle}{\sqrt{2}} \quad (17.5)$$

while if $f(0) \neq f(1)$, meaning we have odd parity, we get

$$(-1)^{f(0)} \frac{|0\rangle - |1\rangle}{\sqrt{2}} \quad (17.6)$$

We can ignore the phase out front, since global phase doesn't affect measurement. Applying another Hadamard gate now gets our quantum state back to the $\{|0\rangle, |1\rangle\}$ basis. If we have even parity, $f(0) = f(1)$, then the output is $|0\rangle$ and if we have odd parity, $f(0) \neq f(1)$, then the output is $|1\rangle$. A complete quantum circuit for the algorithm is given in Figure 17.3.

Note that if we wanted the parity of an n -bit input string, Deutsch's algorithm would let us get that with $\frac{n}{2}$ queries. We simply need to break the string up into $\frac{n}{2}$ blocks of 2 bits each, use Deutsch's algorithm to learn the parity, p_B , of each block B (using $\frac{n}{2}$ queries in total), and then calculate the parity of the p_B 's. This last step doesn't increase the query complexity because it doesn't involve making any additional queries to f . This turns out to be optimal—*any* quantum algorithm to compute the parity of an n -bit string requires at least $\frac{n}{2}$ queries—but we won't prove it in this course.

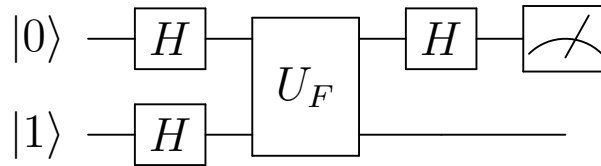


Figure 17.3: Quantum circuit for Deutsch’s algorithm with the additional ancillary qubit used for simulating the phase query shown explicitly.

17.4 Deutsch-Josza Algorithm

Suppose we have a black box that computes a Boolean function $f : \{0, 1\}^n \rightarrow \{0, 1\}$, and suppose we’re promised that f is either a constant function (the output is always 0 or 1, independent of the input), or a balanced function (there are the same number of 0 outputs as 1 outputs). The problem is to decide which. Classically, deterministically, you could solve this problem by examining $2^{n-1} + 1$ values of the function. If all the values match then the function is constant; otherwise the function is balanced. If you want no possibility of error then it’s not hard to see that this is the best you can do. Of course, you can do much better by using random sampling. You’d only need a constant number of queries to get an answer with small probability of error. If all of your samples match, you can guess that the function is constant. Otherwise you know that it’s balanced.

What the Deutsch-Josza algorithm does is to solve the problem perfectly (that is, with zero probability of error) with only one quantum query. That’s something that isn’t possible in the classical world.

Truth is, this speedup still isn’t all that impressive, because the classical probabilistic algorithm is nearly as fast, and would be perfectly fine in practice. Until 1994 or so, non of the quantum speedups that we knew were very impressive!

The quantum circuit for the Deutsch-Josza algorithm is given in Figure 17.4. You’ll begin to notice that some patterns appear a lot in quantum algorithms.

- ▶ You start by putting everything in superposition.
- ▶ You then query a function f —in this case, using a phase query.
- ▶ You then apply a change a basis—in this case, another round of Hadamards.
- ▶ Finally, you measure to get the information you want to know.

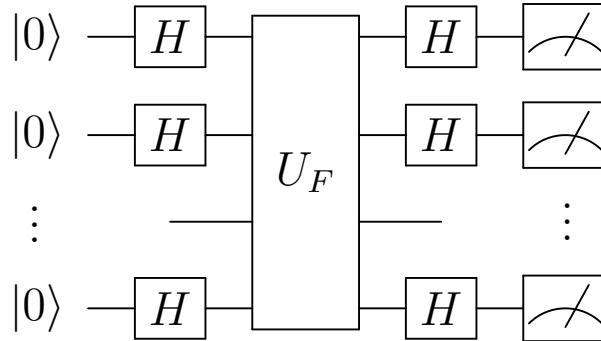


Figure 17.4: Quantum circuit for the Deutsch-Josza algorithm.

If you can't figure out what to do next in a quantum algorithm, a round of Hadamards is usually a good guess!

So, given the circuit for the Deutsch-Josza above (call it \mathcal{C}), *what's the probability of getting back the state $|0 \cdots 0\rangle$?* This is given by $|\langle 0 \cdots 0 | \mathcal{C} | 0 \cdots 0 \rangle|^2$. The first step in the algorithm is to apply a Hadamard gate to each of the n input qubits. As we mentioned above, applying a round of Hadamard gates to each qubit in our circuit is such a common primitive in quantum algorithms that it will be helpful to describe the effect of the transformation on arbitrary standard basis states. The first thing to note is that we can rewrite the effect of the Hadamard gate on a single qubit as

$$H|x\rangle = \frac{|0\rangle + (-1)^x|1\rangle}{\sqrt{2}} \quad (17.7)$$

As such, given an input state $|x\rangle = |x_0, \dots, x_{n-1}\rangle$ the result of applying Hadamard gates to each qubit is

$$\frac{|0\rangle + (-1)^{x_0}|1\rangle}{\sqrt{2}} \otimes \frac{|0\rangle + (-1)^{x_1}|1\rangle}{\sqrt{2}} \otimes \dots \otimes \frac{|0\rangle + (-1)^{x_{n-1}}|1\rangle}{\sqrt{2}}. \quad (17.8)$$

With a bit of algebra this can be simplified to

$$H^{\otimes n}|x\rangle = \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle. \quad (17.9)$$

Here $x \cdot y$ denotes the inner product. This formula essentially says that we pick up a -1 phase for every i such that $x_i = y_i = 1$. Now, coming back to

the Deutsch-Jozsa algorithm, after we Hadamard all n of the qubits and then query the oracle, we get the state

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle. \quad (17.10)$$

Following the second round of Hadamards we get

$$\frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle. \quad (17.11)$$

Rather than simplifying this entire sum, let's take a shortcut by just looking at the amplitude for the state $y = |0 \cdots 0\rangle$. This amplitude is given by

$$\frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{f(x)}. \quad (17.12)$$

If f is constant then the above amplitude is either 1 (if f is always 0) or -1 (if f is always 1). On the other hand, if f is balanced then the amplitude is 0. So when we measure, if we see the outcome $|0 \cdots 0\rangle$ then we know that f is constant, and if we see any other outcome then we know that f is balanced!

Lecture 18: Bernstein-Vazirani and Simon's Algorithm

18.1 The Bernstein-Vazirani Problem

We ended last lecture with the Deutsch-Jozsa problem. In this lecture we'll start with another black-box problem for which quantum algorithms provide an advantage, the **Bernstein-Vazirani Problem**. Here we're given access to a black box function $f : \{0, 1\}^n \rightarrow \{0, 1\}$. We're promised that

$$f(x) = s \cdot x \pmod{2},$$

for some secret string s . The problem is to find s . Classically, you could get an answer one bit at a time by querying all the strings of Hamming weight one. For example, with $n = 4$,

$$f(1000) = s_1$$

$$f(0100) = s_2$$

$$f(0010) = s_3$$

$$f(0001) = s_4$$

But, no classical algorithm can do better than this, since each query can only provide one new bit of information about s , and s has n bits. The Bernstein-Vazirani algorithm, however, solves the problem quantumly using only one query!

18.1.1 Quantum Algorithm

We'll start the quantum algorithm for the Bernstein-Vazirani problem the same way we did for the Deutsch-Jozsa problem, by creating a uniform superposition over all possible input states and then applying a phase query. If our initial

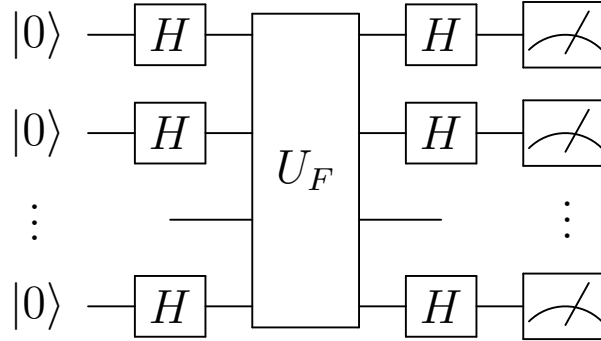


Figure 18.1: Quantum circuit for the Bernstein-Vazirani algorithm. Note that, aside from having a different oracle, the circuit is identical to Figure 17.4

state is the all zero string $|0 \cdots 0\rangle$, then after a round of Hadamards, the state of the system is given by

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle. \quad (18.1)$$

Then, after a phase query, the state of the system is given by

$$|\psi\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{f(x)} |x\rangle = \frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{s \cdot x} |x\rangle. \quad (18.2)$$

The question is *how do we measure the resulting state in a way that gives us information about the secret string s ?* To see how to proceed, let's start by restating the observation originally made in Equations 17.8 and 17.9, that given an input state $|x\rangle = |x_0, \dots, x_{n-1}\rangle$ the effect of applying a Hadamard gate to all n qubits is

$$\begin{aligned} H^{\otimes n} |x\rangle &= \frac{(|0\rangle + (-1)^{x_0} |1\rangle)}{\sqrt{2}} \otimes \frac{(|0\rangle + (-1)^{x_1} |1\rangle)}{\sqrt{2}} \otimes \dots \otimes \frac{(|0\rangle + (-1)^{x_{n-1}} |1\rangle)}{\sqrt{2}} \\ &= \frac{1}{\sqrt{2^n}} \sum_{y \in \{0,1\}^n} (-1)^{x \cdot y} |y\rangle. \end{aligned} \quad (18.3)$$

The key observation is that the state of the system following the phase query has precisely the same form as the state in Equation 18.3 above.

By reversibility, this means that if we Hadamard all the qubits again, we'll change the qubits that picked up a phase (i.e., for which $s_i = 1$) from $|-\rangle$ to

$|1\rangle$ and the qubits that didn't pick up a phase ($s_i = 0$) from $|+\rangle$ to $|0\rangle$:

$$H^{\otimes n} \left(\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} (-1)^{s \cdot x} |x\rangle \right) = |s\rangle \quad (18.4)$$

So, from here we can simply measure the qubits in the standard basis to retrieve $|s\rangle = |s_0, \dots, s_{n-1}\rangle$.

This convenient result isn't a coincidence, you can see that Bernstein and Vazirani designed their problem around what a quantum computer would be able to do!

Don't tell anyone, but this is actually pretty common in this field.

18.2 Simon's Problem

The next algorithm we'll study was discovered in 1994 by Daniel Simon and solves a problem now known as **Simon's Problem**. The story goes that Simon looked at the quantum algorithms coming out and he didn't believe that any of them would give a "serious" speedup. Even the Bernstein-Vazirani problem is easy classically, a classical computer can find the n bits of s with n queries. Sure, the quantum algorithm needs only one query, but it also requires $\mathcal{O}(n)$ gates, so maybe it's not that impressive after all. Simon believed there was a limit that would prevent you from getting a "true" exponential speedup from a quantum computer and he set out to prove it. What he ended up finding instead was that there *is* a true exponential speedup, at least in the black-box setting. As we'll see, this then played a central role in subsequent progress in quantum algorithms, particularly Shor's algorithm which came shortly afterward.

In Simon's problem we're once again given an oracle function f , this time mapping n bits to n bits, $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$. We're promised that there's a secret string $s \neq 0^n$, such that

$$f(x) = f(y) \iff y = x \oplus s \quad (18.5)$$

for all inputs x and y , where \oplus is the bitwise XOR. The problem is to find the secret string s by querying f as few times as possible. Compared to Bernstein-Vazirani, there's more freedom in the choice of function f . In Simon's problem all we require is that f has a "hidden XOR-mask," that is, a subset of bits

such that when you flip the bits in that subset the output is unaffected. *What does this mean?* Let's do an example with 3-bit inputs and with secret string $s = 110$. Let's say we query f a few times and get the following outputs

$$f(000) = 5$$

$$f(110) = 0$$

$$f(001) = 6$$

$$f(111) = 6$$

We're given no information about how to interpret the outputs themselves, so it doesn't really matter whether we think of them as strings, integers, or whatever. The only thing that *does* matter is whether two inputs map to the same output. Since $f(001) = f(111) = 6$, we now know that $s = 001 \oplus 111 = 110$.

This is simple enough with 3 bits, but we're more interested in f 's with, say, 1000-bit inputs. In that case, we claim that finding the secret string is prohibitively expensive classically. *How expensive?* Well, it's not hard to show that any deterministic algorithm needs to query f at least $2^{n-1} + 1$ times, by an argument similar to the one that we used for Deutsch-Jozsa. But once again, the more relevant question is how many queries are needed for a *randomized* algorithm.

We claim that we can do a little bit better with a randomized algorithm, getting down to $\mathcal{O}(2^{n/2})$ queries. This is related to the famous **Birthday Paradox**, which isn't really a paradox so much as a "birthday fact." It simply says that, if you gather merely 23 people in a room, that's enough to have a $\sim 50\%$ probability of getting at least one pair of people who share a birthday. More generally, if there were n days in the year, then you'd need about \sqrt{n} people in the room for a likely birthday collision. At least assuming birthdays are uniformly distributed, in reality they're not exactly, e.g., there are clusters of them about 9 months after major holidays¹. The takeaway here is that the number of pairs of people is what's important and that scales quadratically with the number of people.

The Birthday Paradox is also useful in cryptanalysis. For example, cryptographic hash functions need to make it intractable to find any two inputs x and y with the same hash value, $f(x) = f(y)$. But, by using a "birthday attack"—i.e., repeatedly choosing a random input x , then comparing $f(x)$ against $f(y)$ for *every* previously queried input y —we can find a collision using a number of queries that scales only like the square root of the size of f 's range. This is quadratically faster than one might have expected naively.

¹But it can be shown that any nonuniformities only make collisions *more* likely.

Whatever other structure it has, Simon's problem involves a two-to-one function in which we're looking for a collision, so it also admits a birthday attack. Roughly speaking, given two randomly-chosen inputs x and y , we'll observe $f(x) = f(y)$ with probability $p \approx 2^{-n}$ and while these events aren't quite independent between the various (x, y) pairs they're nearly so. If we want to observe a collision with high probability then we need to query $f \sim 2^{n/2}$ times.

18.2.1 Classical Lower Bound

Is there a better classical algorithm? Let's prove that the answer is no. We'll use an **Adversary Argument**, essentially asking "If my worst enemy got to choose f , what would they do?" Presumably they would choose a secret string s uniformly at random among all possible s 's, to make it as hard as possible to find an underlying structure in f . Then, perhaps, they'd choose a random f among all those consistent with that choice of s .

Now, once we fix such a strategy for the adversary we can assume without loss of generality that the algorithm is deterministic. This is because any randomized algorithm can be thought of as just a probabilistic mixture of deterministic algorithms, and there must be at least one algorithm in the mixture that does at least as well as the average! This observation—together with the complementary observation that *all* randomized lower bounds can be proved in this way—is sometimes referred to as Yao's minimax principle.

So, let the deterministically queried inputs be x_0, x_1 , etc. Then the question is *what information can we derive about s after the first t queries?*

If after t queries we've found a collision pair, $f(x_i) = f(x_j)$ for some $i \neq j$, then we're done since we know that $s = x_i \oplus x_j$. So let's assume that hasn't happened yet. In that case, all we can conclude about s is that $s \neq x_i \oplus x_j$ for every $i \neq j$ pair queried thus far. This rules out at most t^2 possible values of s , with all the other possibilities remaining equally likely given what we've seen so far (i.e., having equal posterior probabilities). It follows that unless we observe a collision, narrowing the possibilities down to a single s requires $\Omega(2^{n/2})$ queries. The probability that we observe a collision by the t^{th} query is only

$$\frac{\binom{t}{2}}{2^n - 1},$$

so, by the union bound, with high probability we won't observe any collisions at all until we've made $\sim 2^{n/2}$ queries. And that's the adversary argument.

18.2.2 Quantum Algorithm

Amazingly, in contrast to the $\Omega(2^{n/2})$ classical lower bound, we can solve Simon's problem on a quantum computer using only $\mathcal{O}(n)$ queries to f . The quantum algorithm that does this is known as **Simon's Algorithm**. The quantum circuit is given in Figure 18.2

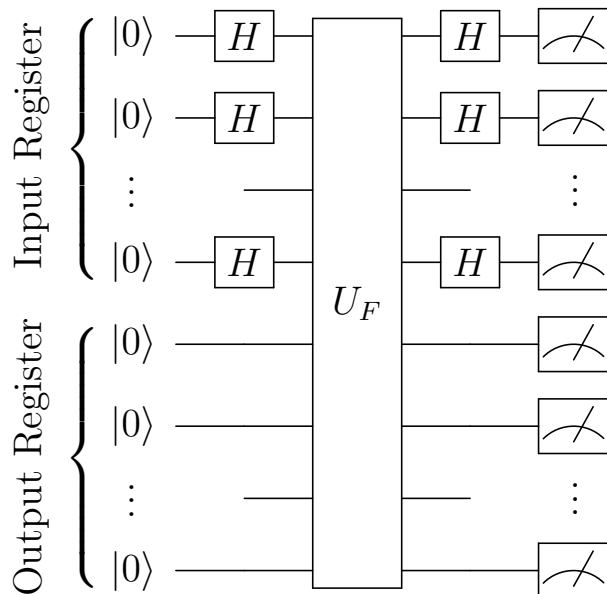


Figure 18.2: Quantum circuit for the Deutsch-Josza algorithm.

The algorithm follows a now-familiar pattern:

- ▶ Start with $2n$ qubits initialized to the $|0 \cdots 0\rangle$ state. The first n will be the input register and the second n will be the answer register.
- ▶ Hadamard the first n qubits.
- ▶ Query f using an XOR oracle.

This yields the state

$$\frac{1}{\sqrt{2^n}} \sum_{x \in \{0,1\}^n} |x\rangle |f(x)\rangle. \quad (18.6)$$

This time the function f has a large output so we need to write out its values in a separate n -qubit answer register rather than just encoding it into the phase. But, it's important to note that the answers themselves aren't what we care about! We're only writing them out because by doing so, we create a

desired interference pattern in the input register. Indeed, at this point in the algorithm we could simply discard the answer register, or do anything else we liked with them. For pedagogical simplicity let's assume we now *measure* the answer register, and let's assume that the result of the measurement is $|w\rangle$.

Keeping track of all of the w 's we could have seen would've resulted in a mixed state. Instead, we're just conditioning on a particular $|w\rangle$.

By the partial measurement rule we're left with an equal superposition over all the different inputs that are consistent with the value of w that we observed. In Simon's problem, there are necessarily two such inputs. In other words, we're left with a superposition

$$\frac{|x\rangle + |y\rangle}{\sqrt{2}} \text{ such that } f(x) = f(y) = w \quad (18.7)$$

By the Simon promise, this means in particular that $s = x \oplus y$. *So what is this state good for?* First, observe that if we could just measure the state twice, then with high probability we'd get both x and y . Bitwise-XORing the two strings would then give us the secret string s and we'd be done!

Alas, in quantum mechanics we only get one chance to measure a state, so we'll see either x or y , which tells us nothing about s . We could of course repeat the whole algorithm from the beginning, but if we did then with overwhelming probability we'd get a different w corresponding to a new pair.

So, we'll need to be more clever—although not that much more clever! In particular, let's see what happens if we measure the state in Equation 18.7 in the Hadamard basis. For starters, we know from Equation 18.3 that Hadamarding all n qubits in the standard basis state $|x\rangle$ maps it to

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

and likewise, Hadamarding all n qubits of $|y\rangle$ gives

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

By linearity, this means that applying $H^{\otimes n}$ to an equal superposition of

$|x\rangle$ and $|y\rangle$ must give

$$H^{\otimes n} \left(\frac{|x\rangle + |y\rangle}{\sqrt{2}} \right) = \frac{1}{\sqrt{2^{n+1}}} \sum_{z \in \{0,1\}^n} [(-1)^{x \cdot z} + (-1)^{y \cdot z}] |z\rangle. \quad (18.8)$$

Now we'll measure this state in the standard basis. *Which z 's could we get when we do so?* For a given z to be observed, it must have a nonzero amplitude. This means that $(-1)^{x \cdot z}$ and $(-1)^{y \cdot z}$ must be equal, which occurs if and only if $x \cdot z = y \cdot z \pmod{2}$. Or, rewriting this equation a bit:

$$\begin{aligned} x \cdot z + y \cdot z &= 0 \pmod{2} \\ (x \oplus y) \cdot z &= 0 \pmod{2} \\ s \cdot z &= 0 \pmod{2} \end{aligned} \quad (18.9)$$

So, what we get when we measure is an n -bit string z , chosen uniformly at random from among all of the 2^{n-1} strings whose inner product with s is 0. In other words, we haven't yet learned s itself, but we've learned a bit of information about s , which I hope you'll grant is something! *What if we really want s itself?* In that case, we can just repeat Simon's algorithm over and over, starting from the beginning each time! This will give us a collection of strings $\{z_0, \dots, z_{k-1}\}$, which are selected uniformly random and independent of each other, and which all have even inner products with s :

$$\begin{aligned} s \cdot z_0 &= 0 \pmod{2} \\ s \cdot z_1 &= 0 \pmod{2} \\ &\vdots \\ s \cdot z_{k-1} &= 0 \pmod{2} \end{aligned} \quad (18.10)$$

Now that we've got these equations, *what should we tell a classical computer to do with them?* Well, suppose $k = n + c$, where c is some large constant. Then we now have a collection of $n + c$ linear equations in n unknowns over a finite field with two elements. We can solve this system of equations efficiently using a classical computer.

Using an algorithm called "Run Matlab." Or Gaussian elimination, taking $\mathcal{O}(n^3)$ time. Or, if you're feeling fancy—and impractical, since the constant factor overheads are monumental—the fastest known algorithm for solving linear systems takes $\mathcal{O}(n^{2.373})$ time.

It's not hard to do a probabilistic analysis showing that after we've seen slightly more than n equations, with overwhelming probability we'll be left with a system of equations that has exactly two solutions. Namely, 0^n and s itself. We can throw away 0^n , because we assumed $s \neq 0^n$. So that leaves us with s .

So, with $\mathcal{O}(n)$ queries along with a polynomial amount of additional classical computation we can find s , in sharp contrast to the provably exponential number of queries needed to solve Simon's problem classically. A few additional questions:

Does Simon's algorithm have a deterministic counterpart? Yes, one can modify the algorithm so that it succeeds with certainty rather than "merely" overwhelming probability. We won't go into the details here.

Why doesn't this just prove that quantum algorithms are better? It's sort of tricky to translate Simon's algorithm from the black-box setting into the "real-world." To get a speedup over classical computing in terms of the sheer number of gates or computational steps, we'd need some small circuit to compute a function f that was actually like our magical Simon function.

To illustrate, $f(x) = Ax$ for some rank- $(n-1)$ Boolean matrix A would satisfy the Simon promise. But, the difficulty in getting a quantum speedup this way is that once we pin down the details of how we're computing f —in our example, as soon as we give the actual matrix A —we then need to compare against classical algorithms that know those implementation details as well. As soon as we reveal the innards of the black box, the odds of an efficient classical solution become much higher! In our example, if we knew the matrix A then we could solve Simon's problem in classical polynomial time just by calculating A 's nullspace. More generally, no one to this day has found a straightforward "application" of Simon's algorithm, in the sense of a class of efficiently computable functions f that satisfy the Simon promise and for which any classical algorithm plausibly needs exponential time to solve Simon's problem when the algorithm is given the implementation details of f .

The story goes that Daniel Simon wrote a paper about this theoretical black-box problem with an exponential quantum speedup and the paper got rejected. But there was one guy on the program committee who was like, "hey, this is interesting." He figured that if you changed a few aspects of what Simon was doing, you could get a quantum algorithm to find the periods of periodic functions, which would in turn let you do all sorts of fun stuff. That guy was Peter Shor and the algorithm he invented will be the focus of our next three lectures.

Lecture 19: RSA and Shor's Algorithm

19.1 RSA Encryption

In this lecture we'll see Shor's algorithm. Given a positive integer N , which we'll assume for simplicity is a product of two primes p and q , this algorithm lets you find p and q using only about $\mathcal{O}(\log^2(N))$ steps. Shor's algorithm captured the world's attention because **RSA**, one of the most widely-used public-key encryption methods, relies on the assumption that factoring is hard. So, before we start in on Shor's algorithm, which will take a few lectures to cover completely, let's briefly review RSA.

The basic idea is that some website, say Amazon, wants you to be able to send them messages that only they can decrypt (your credit card number, for example). But, they've never met with you in private to agree on a secret encryption key, so private-key cryptography is off the table. And we don't have the hardware for quantum key distribution! So, what Amazon does instead is find two large primes p and q (say a thousand digits each), which they keep secret. Amazon then multiplies them to get $N = pq$, which they publish to the world. The fact that Amazon can efficiently pick two huge random prime numbers p and q and know for sure that they're prime is already not quite obvious, but it follows from some classical number theory that we won't go into.

Now, you can encrypt a message using the public key, N . If your plaintext message is x , then in the simplest version of RSA, your encrypted message would just be $x^3 \bmod N$. Given that encrypted message along with knowledge of the prime factors p and q there's an efficient algorithm for Amazon to recover x —it's again some basic number theory that we won't go into right now, although we'll see aspects of it later. The key idea is that if you know p and q , then you also know the order of the *multiplicative group* modulo N , and that knowledge lets you do things like efficiently take cube roots modulo

N . By contrast, an eavesdropper who doesn't know p and q , but only knows N , seems to face an exponentially hard problem in recovering the plaintext—though it's been proven neither that factoring is hard, nor even that breaking RSA is necessarily as hard as factoring.

Some number theorists conjecture that factoring is in P , or profess agnosticism. The problem of factoring's complexity has only been seriously worked on for, like, 40 years. In any case, if you can factor, then you can break RSA, and that certainly provides more than enough reason to be interested in the complexity of factoring.

The naïve algorithm to factor N is trial division, which in the worst case requires you to test all possible divisors up to \sqrt{N} (every divisor greater than \sqrt{N} must have an accompanying divisor that's smaller). We call this an “exponential-time” algorithm since the running time is exponential in $n = \log(N)$, which is the number of digits needed to specify N . Number theorists have discovered several faster algorithms. The Quadratic Field Sieve, from 1981, runs in roughly $2^{\mathcal{O}(\sqrt{N})}$, a milder exponential. The Number Field Sieve brings that down to $2^{\mathcal{O}(N^{1/3})}$, though its correctness depends on the proof of a yet-unproven conjecture.

This is why 512-bit, 768-bit, and maybe even 1024-bit encryption aren't quite secure anymore. They can be cracked using known algorithms given sufficient time and money for hardware. In the Snowden documents there's evidence of the NSA allocating money for this sort of thing. In a way that's almost reassuring to people who worry that the NSA can break anything...

Another public-key cryptosystem in widespread use is **Diffie-Hellman**, which is based on a different problem called discrete logarithms. While we won't cover this system in this lecture, it turns out that Shor's algorithm also solves the discrete-log problem in polynomial time, thereby breaking Diffie-Hellman as well.

No one has shown that factoring and discrete log are necessarily related, e.g. by giving a reduction between them. In practice, though, advances in solving one problem almost always seem to lead to advances in solving the other

in short order.

We now know that both RSA and Diffie-Hellman were first discovered in secret—by the mathematician Clifford Cocks for the former and by James H. Ellis, Clifford Cocks and Malcolm J. Williamson for the latter—at GCHQ (the British NSA) before they were rediscovered in public about a few years later.

19.2 Period Finding

What Shor's algorithm really does, under the hood, is solve a problem called **Period-Finding**. One of Shor's key observations was that, for classical number theory reasons having nothing to do with quantum mechanics, a fast algorithm for period-finding leads to a fast algorithm for problems like factoring and discrete logarithm. Period-finding is a black-box problem similar in some respects to Simon's Problem. In period-finding, we're given oracle access to a function $f : \mathbb{N} \rightarrow \mathbb{N}$, and promised that there's some secret integer $s > 0$ such that for all x and y

$$f(x) = f(y) \iff s|(y - x). \quad (19.1)$$

We call s the period of the function f . The problem is to find s .

How many queries to f do we need to solve period-finding classically? Let's assume that $s \sim 2^n$ (i.e., that s is an n -bit integer) and give our answer in terms of n . Observe that once you find a pair x and y such that $f(x) = f(y)$, you then know that s divides $y - x$ and are very close to solving the problem. Indeed, if you found a few such collisions you could just take their greatest common divisor $\gcd(x_1 - y_1, x_2 - y_2, x_3 - y_3, \dots)$. We won't give the analysis here, but after not too many collisions the odds are high that this will yield the period.

Incidentally, how do we get the gcd of two integers in polynomial time? We use **Euclid's GCD Algorithm**—possibly the oldest interesting polynomial-time algorithm in history! To find the gcd of x and y (with $x > y$), we find q and r that satisfy

$$x = qy + r, \quad (19.2)$$

such that qy is the greatest multiple of y less than x . This means that $y > r$. Then, we find the gcd of y and r and we keep recursing in this way until $r = 0$. The size of the numbers involved in each recursive step goes down by a constant factor each time, which means the whole algorithm runs in time linear in n , the number of digits of x and y .

OK, but how do we find the collisions classically? This is the birthday paradox all over again. Recall from the last lecture that something like $\sim 2^{n/2}$ queries to f are both necessary and sufficient.

*This is **still** a huge number of queries, if (say) $n = 2000$.*

19.2.1 Factoring to Period-Finding Reduction

The first part of Shor's algorithm is a purely classical reduction from factoring to period-finding. We'll talk about this reduction first, before moving on to the quantum algorithm for efficiently solving the period-finding problem.

Why is factoring reducible to period finding? The main connection between the two is the *multiplicative group modulo N* , typically denoted \mathbb{Z}_N^\times . This is a finite abelian group—that is, a finite set with a commutative, associative, invertible multiplication operation—that's one of the most basic examples of a group in all of math. The multiplicative group modulo N is given by the set of positive integers less than N which are *relatively prime* to N , meaning that the only common factor they share with N is 1. The group operation is multiplication mod N . For a prime p , the multiplicative group consists of the set $\{1, 2, \dots, p-1\}$, with the group operation being multiplication modulo p . The restriction to integers relatively prime to N is essential when N is composite, since otherwise you won't have a multiplicative inverse. For example, when $N = 15$ the multiplicative group mod N consists of the 8-element set $\{1, 2, 4, 7, 8, 11, 13, 14\}$. In general, the size of this group, given $N = pq$, is going to be $|\mathbb{Z}_N^\times| = (p-1)(q-1)$, since that's how many positive integers less than N are relatively prime to N . For example, when $N = 15$, we saw that the size of the group is $(5-1)(3-1) = 8$.

An extremely useful fact about finite groups G is that, for any element x , we have $x^{|G|} = 1$. This has a few corollaries, such as **Fermat's Little Theorem**, which states that for all primes p and integers $x \in \{1, \dots, p-1\}$

$$x^{p-1} \equiv 1 \pmod{p} \tag{19.3}$$

Another important corollary is a generalization of Fermat's little theorem called **Euler's Theorem**, which says that for all primes p and q , and for integers x relatively prime to p and q , the following holds:

$$x^{(p-1)(q-1)} \equiv 1 \pmod{pq} \tag{19.4}$$

Euler's Theorem is super important in RSA encryption as well.

Euler's Totient Function $\varphi(N)$ returns the order of the multiplicative group mod N , which is the number of integers from 1 to N that are relatively prime to N . For example, if p and q are prime, then $\varphi(p) = p - 1$ and $\varphi(pq) = (p - 1)(q - 1)$. If $n = pq$, then we can rewrite Euler's theorem from above in terms of the totient function:

$$x^{\varphi(N)} \equiv 1 \pmod{N} \quad (19.5)$$

Why is this important? Well, let's say we want to factor some number $N = pq$, a product of distinct primes. Then here's an approach: pick an x such that $\gcd(x, N) = 1$.

Such x 's are easy to find. Indeed, in the rare event that we pick an x and it happens that $\gcd(x, N) > 1$, we can run Euclid's algorithm on x and N to factor N right then and there!

Shor's algorithm relies crucially on the properties of the modular exponentiation function,

$$f(r) = x^r \pmod{N}. \quad (19.6)$$

What can we say about this function? First of all, how hard is it to compute? A naive approach would use $r-1$ multiplications, but r can be large and so this can end up being exponential in $n = \log(N)$. But there's a much, much faster approach called **Repeated Squaring**. It's best illustrated with an example. Say we want to calculate $13^{21} \pmod{15}$. We could calculate $13 \times 13 \times \cdots \times 13 \pmod{15}$ by alternating multiplication with reducing mod 15, but that's still 20 multiplications. Instead, let's rewrite it as a product of 13 raised to various powers of 2:

$$13^{16} \times 13^4 \times 13 \pmod{15}.$$

We can further rewrite this as

$$\left(\left(\left((13^2)^2 \right)^2 \right)^2 \right)^2 \times (13^2)^2 \times 13 \pmod{15}$$

This may be ugly, but it requires considerably fewer multiplications (6 in total), an advantage that grows rapidly as the exponent increases. Indeed, the total time needed is polynomial in n .

Once again, repeated squaring also plays a central role in RSA decryption. Ironically, many of the same number theory facts that led to RSA also allow lead to Shor's algorithm, which breaks RSA.

OK, so $f(r) = x^r \pmod N$ is efficiently computable. It's also, clearly, a periodic function, with a period that divides $\phi(N)$, the order of \mathbb{Z}_N^\times . *What could we learn by figuring out its period?* Here's the key point: we claim that *finding the period of f will let us factor N . Why?* First, some intuition. Suppose we were able to learn $\varphi(N)$, the order of the multiplicative group mod N . Then we'd surely be able to factor N into pq . This is because

$$\varphi(N) = (p-1)(q-1) = pq - p - q + 1$$

So, if we know $\varphi(N)$ we now know both $N = pq$ and $p+q$, and we can use the quadratic formula to solve for p and q themselves. Unfortunately, this doesn't quite work with Shor's algorithm, because the period of f might not equal $\varphi(N)$, the most we can say is that the period *divides* $\varphi(N)$.

Here's what we'll do instead. Let's pick a random x relatively prime to N , and find the period s of $f(r) = x^r \pmod N$. We then have that $x^s \equiv 1 \pmod N$. Now, let's imagine we're lucky and s is even (which intuitively should happen maybe half the time?). In that case we can write

$$x^s - 1 = (x^{s/2} - 1)(x^{s/2} + 1) \equiv 0 \pmod N. \quad (19.7)$$

In other words, the product $(x^{s/2} - 1)(x^{s/2} + 1)$ is an integer multiple of N . Now, suppose we get lucky a second time, and neither $x^{s/2} - 1$ nor $x^{s/2} + 1$ is itself a multiple of N . Then that means *we've learned a factor of N* . We just compute $\gcd(x^{s/2} - 1, N)$, which will give us either p or q . This is because if neither $x^{s/2} - 1$ nor $x^{s/2} + 1$ is a multiple of N , then in order to satisfy Equation 19.7, one term must be a multiple of p and the other a multiple of q .

Furthermore, both of these "imagine we get lucky" steps can be shown to happen with a constant probability over the choice of x by using a little number theory that we won't go into here. The precise statement is for any N , if x is randomly chosen from the numbers relatively prime to N , then with probability at least $\frac{3}{8}$:

- s is even.

- ▶ Neither $x^{s/2} - 1$ nor $x^{s/2} + 1$ is a multiple of N .

This gives us a plan of attack for factoring N . The key remaining problem—and the one we'll use quantum mechanics to solve—is finding the period of f .

19.2.2 Quantum Algorithm for Period-Finding

We'll now give a quantum algorithm that solves the black-box problem of period-finding in time polynomial in $n = \log(N)$. We'll then apply that algorithm to find the period of the function $f(r) = x^r \pmod N$, for a randomly chosen value of x relatively prime to N . That is, we'll use f to “instantiate” the black box. By the previous discussion, f can be computed in polynomial time using repeated squaring. Better yet, the ability to find its period implies the ability to factor N .

Recall that for any given x the probability that it satisfies the constraints we need in order to use the period of f to factor a given N is $\sim \frac{3}{8}$, so we might need to try several different values of x until we find one that works.

The first step in the quantum period finding algorithm is to make an equal superposition over all nonnegative integers less than some upper bound Q ,

$$\frac{1}{\sqrt{Q}} \sum_{r=0}^{Q-1} |r\rangle |f(r)\rangle, \quad (19.8)$$

with each integer written in its binary representation. For technical reasons we'll explain in a later lecture, we set Q to be a power of 2 of order N^2 . We can prepare this state by Hadamarding the $\log(Q)$ qubits in the input register, then querying f and writing the output into the $\log(Q)$ qubit of the answer register (with uncomputing to get rid of garbage).

Unlike with Simon's algorithm, in Shor's algorithm U_f is not just an abstract black box. We can find an actual quantum circuit to implement U_f , because f is just the modular exponentiation function.

By using the repeated squaring trick, we can create actual circuit for U_f that maps $|r\rangle |0 \cdots 0\rangle$ to $|r\rangle |f(r)\rangle$ out of a network of $\text{polylog}(N)$ Toffoli gates.

Just like with Simon's algorithm, we won't care at all about the actual value of $f(r)$, but only about the effect that computing $f(r)$ has on the $|r\rangle$ register. So for pedagogical purposes, we'll immediately measure the $|f(r)\rangle$ register and then discard the result.

What's left in the input register? By the partial measurement rule, what's left is an equal superposition over all the possible r 's that could've led to the observed value $f(r)$. Since f is periodic with a secret period s , these values will differ from each other by multiples of s . In other words, we now have the state

$$|\psi\rangle = \frac{|r\rangle + |r+s\rangle + |r+2s\rangle + \cdots + |r+(L-1)s\rangle}{\sqrt{L}} \quad (19.9)$$

where L is an appropriately chosen normalization constant.

The central challenge of Shor's algorithm is to measure the above state in a way that reveals useful information about the period s .

Just like in Simon's Algorithm, if we could measure the state multiple times in the standard basis (without collapsing it), then we could take the gcds of the differences to find s . But we can't do that! We can repeat the whole algorithm from the beginning, but if we do then we'll almost certainly end up with a different offset r , preventing useful comparisons.

This simply means that—just like with everything else in quantum computing—to see a speedup we'll have to exploit the magic of minus signs: interference, cancellations, change of basis, whatever term you want to use. *How do we change the basis to one that reveals the period, and moreover do so efficiently?*

In the next lecture we'll see how to do just that, using the Quantum Fourier Transform!

Lecture 20: Quantum Fourier Transform

In the previous lecture we started in on Shor’s algorithm, a quantum algorithm that can factor N into its prime factors p and q in polynomial time by reducing the problem to period-finding. Now we’ll see how to solve period-finding in polynomial time. Before we do, though, let’s address a conceptual clarification: *Is Shor’s algorithm provably faster than any classical algorithm for the same task?*

If by “Shor’s algorithm,” we mean the period-finding core of the algorithm, then the answer is yes. If, on the other hand, we think of Shor’s algorithm as a way to factor integers, then the speedup remains conjectural. Indeed it has to be, because no one has even proven that $\mathbf{P} \neq \mathbf{NP}$ —and if $\mathbf{P} = \mathbf{NP}$, then of course factoring is classically easy. The way to reconcile these two statements is simply to observe that there are many ways to factor a number besides by reducing factoring to period-finding. In fact, the best known classical factoring algorithms—the Quadratic Field Sieve and Number Field Sieve mentioned in the last lecture—do much better than the naïve classical birthday attack by exploiting additional structure in the factoring problem. The most we can currently prove is that Shor’s algorithm achieves an exponential speedup over any classical factoring algorithm that works via the last lecture’s reduction to period-finding.

20.1 Quantum Fourier Transform

We left off last time with our quantum state in the form

$$|\psi\rangle = \frac{|r\rangle + |r+s\rangle + |r+2s\rangle + \cdots + |r+(L-1)s\rangle}{\sqrt{L}}$$

Now we’ll see how to measure this state to extract useful information about

the period s . In science and engineering, any time you have a periodic signal and you're trying to extract its period there's an essential tool used called the **Fourier Transform**. There are many types of Fourier transforms: continuous, Boolean, etc. For us, though, the Q -dimensional **Quantum Fourier Transform** or **QFT** will be the $Q \times Q$ matrix F_Q defined as follows:

$$(F_Q)_{i,j} = \langle i | F_Q | j \rangle = \frac{\omega^{ij}}{\sqrt{Q}} \quad (20.1)$$

where $\omega = e^{2\pi i/Q}$ is a Q^{th} root of unity. Here's some useful intuition for how the Fourier transform works. In graduate school you can easily fall into a 26-hour-per-day cycle. So, one day you wake up at 8am, the next day you wake up at 10am, then 12pm, and so forth so that if nothing interrupts you, you cycle all the way around. Suppose you've fallen into such a cycle and you want to figure out how long the cycle is without doing any complicated calculations like subtraction.

What you can do is install a series of clocks in your room, each tracking "days" of different lengths. So, you'd have a 23-hour clock, a 24-hour clock, a 25-hour clock, etc. . . In addition, you install a bulletin board below each clock and place a single thumbtack in its center. Now, every time you wake up you go to each clock and move the thumbtack one inch in the direction the hour hand points.

What will happen if you keep doing this, week after week? If you're really keeping 26-hour days, then the thumbtack corresponding to the 26-hour clock will always move in the same direction. This is constructive interference! And the same is true for the 13-hour clock (as well as the 1-hour and 2-hour clocks). All the others—the 23-hour clock, the 24-hour clock, etc.—will have the thumbtack move around, sometimes one direction, sometimes another, so that it eventually returns to the origin.

The Quantum Fourier Transform is essentially this, but with quantum-mechanical amplitudes instead of thumbtacks.

There are two questions we need to answer here:

- ▶ *How do we implement the Quantum Fourier Transform using a small quantum circuit?*
 - Since it's a $Q \times Q$ matrix, it's not obvious whether we can do it using a circuit with only $\text{polylog}(Q)$ gates.
- ▶ *Once we've applied the QFT and measured, how do we make sense of the outcome?*
 - Complications can arise because in all likelihood, the period s won't evenly divide Q .

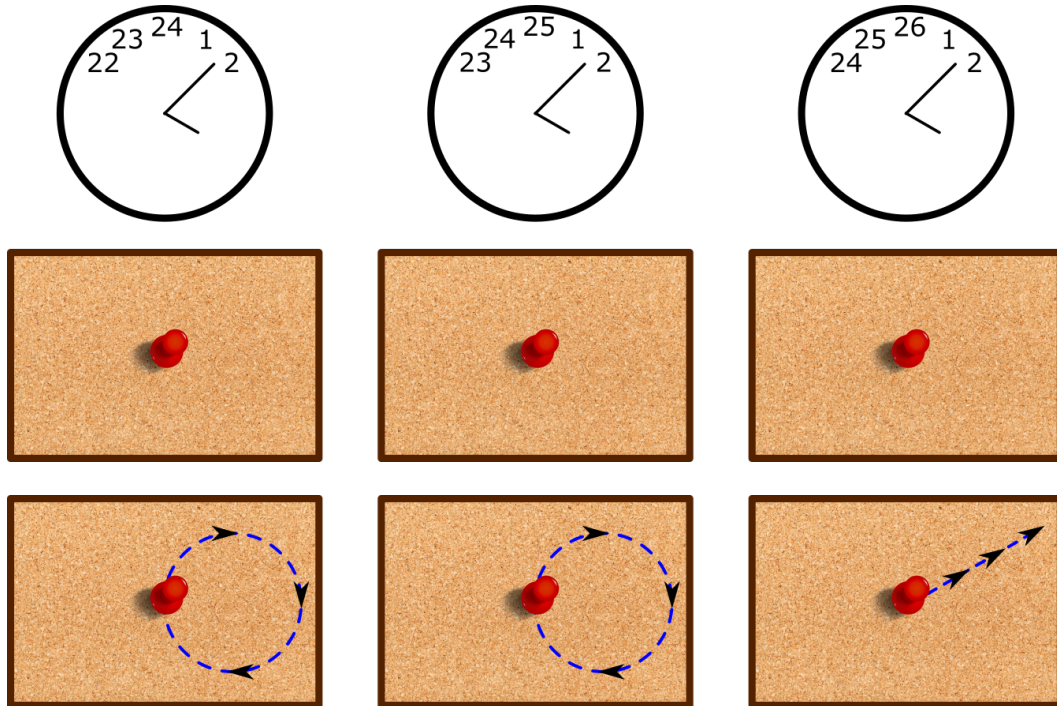


Figure 20.1: A series of clocks, each corresponding to “days” of different lengths. A thumbtack below the clock is shifted in the direction of the hour hand each morning upon waking up. Only the thumbtack corresponding to the correct length “day”—as well as “days” whose length divides the correct day length—will cause the thumbtack to move in the same direction each time. This is an example of constructive interference.

20.1.1 Implementing the QFT

To figure out how to implement the QFT, let's look at some examples of QFT matrices for a few different sizes.

$$F_2 = H = \frac{1}{\sqrt{2}} \begin{bmatrix} \omega^0 & \omega^0 \\ \omega^0 & \omega^1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (20.2)$$

$$F_4 = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega^1 & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega^1 & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^0 & \omega^2 \\ 1 & \omega^3 & \omega^2 & \omega^1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{bmatrix} \quad (20.3)$$

$$F_8 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \omega^1 & \omega^2 & \omega^3 & \omega^4 & \omega^5 & \omega^6 & \omega^7 \\ 1 & \omega^2 & \omega^4 & \omega^6 & 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^1 & \omega^4 & \omega^7 & \omega^2 & \omega^5 \\ 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 & 1 & \omega^4 \\ 1 & \omega^5 & \omega^2 & \omega^7 & \omega^4 & \omega & \omega^6 & \omega^3 \\ 1 & \omega^6 & \omega^4 & \omega^2 & 1 & \omega^6 & \omega^4 & \omega^2 \\ 1 & \omega^7 & \omega^6 & \omega^5 & \omega^4 & \omega^3 & \omega^2 & \omega^1 \end{bmatrix} \quad (20.4)$$

As in Equation 20.1, the ω 's in the examples above are Q^{th} roots of unity where Q is the dimension of the matrix. We could design an algorithm to apply these matrices by brute force, but there's a better way. This method is related to one of the most widely used classical algorithms, the **Fast Fourier Transform** or **FFT**.

Suppose we have a vector of length Q , and we want to apply a $Q \times Q$ matrix A to it. In general this requires us to do the full matrix-vector multiplication which takes $\sim Q^2$. However, if we know that A is the Fourier transform, then the FFT lets us apply it in only $\mathcal{O}(Q \log(Q))$ steps, by exploiting regularities in the Fourier matrix.

What regularity is there in the Fourier matrix? Look at F_4 . If we swap the second and third columns, we get:

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

Notice how the matrix can be broken up into 4 blocks, each either equal to H or related to H by the application of a diagonal matrix. In fact if we define the matrix B as

$$B = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix} \quad (20.5)$$

then we can rewrite the transformed F_4 matrix above as

$$\frac{1}{\sqrt{2}} \begin{bmatrix} H & BH \\ H & -BH \end{bmatrix} \quad (20.6)$$

You can do a similar procedure for F_8 , moving all the odd columns to the left and the even columns to the right. Simplifying the ω 's, we get

$$\frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i & \omega & \omega^3 & \omega^5 & \omega^7 \\ 1 & -1 & 1 & -1 & i & -i & i & -i \\ 1 & -i & -1 & i & \omega^3 & \omega & \omega^7 & \omega^5 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & i & -1 & -i & \omega^5 & \omega^7 & \omega & \omega^3 \\ 1 & -1 & 1 & -1 & -i & i & -i & i \\ 1 & -i & -1 & i & \omega^7 & \omega^5 & \omega^3 & \omega \end{bmatrix}$$

Which we can rewrite as

$$\frac{1}{\sqrt{2}} \begin{bmatrix} F_4 & BF_4 \\ F_4 & -BF_4 \end{bmatrix} \quad (20.7)$$

where the matrix B is now given by

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \omega & 0 & 0 \\ 0 & 0 & \omega^2 & 0 \\ 0 & 0 & 0 & \omega^3 \end{bmatrix}. \quad (20.8)$$

You can work out why it happens on your own, but it turns out that we can define F_Q in terms of this nesting recurrence:

$$F_Q = \frac{1}{\sqrt{2}} \begin{bmatrix} F_{Q/2} & B_{Q/2}F_{Q/2} \\ F_{Q/2} & -B_{Q/2}F_{Q/2} \end{bmatrix}, \quad (20.9)$$

where we've now defined the generalization of the diagonal B matrix above, $B_{Q/2}$, as (omitting the zeros on the off-diagonal)

$$\begin{bmatrix} 1 & & & & & \\ & \omega & & & & \\ & & \omega^2 & & & \\ & & & \ddots & & \\ & & & & \ddots & \\ & & & & & \omega^{\frac{Q}{2}-1} \end{bmatrix}. \quad (20.10)$$

If we let $C(F_Q)$ be the number of steps needed to apply F_Q , we get the recurrence relation

$$C(F_Q) \leq 2C(F_{Q/2}) + \mathcal{O}(Q), \quad (20.11)$$

for which the solutions is $C(F_Q) = \mathcal{O}(Q \log(Q))$.

In the quantum case, we're actually interested in the unitary transformation $|\psi\rangle \rightarrow F_Q |\psi\rangle$, where $|\psi\rangle$ is a quantum state of $\log_2(Q)$ qubits. In one sense, our new goal is less ambitious since we don't need the result vector written explicitly in memory anywhere; it only needs to be encoded implicitly in a vector of amplitudes. In another sense, though, our new goal is more ambitious, since we now want to apply a Fourier transform in time polynomial in only $\log(Q)$.

So how do we do it? We can use the same recursion that we used for the FFT, *plus* the additional observation that the recursion behaves "linearly" with respect to quantum states. Let's think of our $\log(Q)$ qubits as representing an integer from 0 to $Q-1$ in binary notation and let's order the bits of that integer from most to least significant. But let's also reorder the bits, so that the one that was previously the least significant is now the most significant (the analogue of reordering the columns in our recursion for F_Q). Then we can try applying the circuit $F_{Q/2}$ to the "first half" of the number. In other words, we apply $F_{Q/2}$ to all but the (now) most significant bit. This corresponds to applying the matrix (in block notation)

$$F_{Q/2} \otimes I_2 = \begin{bmatrix} F_{Q/2} & 0 \\ 0 & F_{Q/2} \end{bmatrix} \quad (20.12)$$

To get a B in the bottom right quadrant, we can apply a controlled- B gate using the most significant bit as the control. A controlled- B gate can be written in the form

$$\begin{bmatrix} I & 0 \\ 0 & B \end{bmatrix}. \quad (20.13)$$

So after we apply the controlled- B gate, we have

$$\begin{bmatrix} I & 0 \\ 0 & B \end{bmatrix} \begin{bmatrix} F_{Q/2} & 0 \\ 0 & F_{Q/2} \end{bmatrix} = \begin{bmatrix} F_{Q/2} & 0 \\ 0 & BF_{Q/2} \end{bmatrix}. \quad (20.14)$$

We can implement B using a linear number of gates, because it simply amounts to the following:

- ▶ If the most significant bit is 1, then do a rotation by some angle θ
- ▶ If the second bit is 1, then do a rotation that's half as big
- ▶ If the third bit is 1, then do a rotation that's a quarter as big
- ▶ etc...

By the time you reach the last couple of bits, the rotation is exponentially small.

When they first learn about Shor's algorithm, some people object that it's "unphysical," since there's no practical way to apply such tiny rotations. But it turns out that the exponentially small rotations don't matter for the algorithm—indeed, there's a theorem that says that you can just omit these tiny rotations. Doing so even improves the size of the quantum circuit that implements F_Q , from $\mathcal{O}(\log^2(Q))$ to $\mathcal{O}(\log(Q)\log\log(Q))$.

The final step to get the matrix we want is a Hadamard gate applied to the most significant bit,

$$H \otimes I_{Q/2} = \frac{1}{\sqrt{2}} \begin{bmatrix} I_{Q/2} & I_{Q/2} \\ I_{Q/2} & -I_{Q/2} \end{bmatrix}, \quad (20.15)$$

which results in the final matrix

$$\frac{1}{\sqrt{2}} \begin{bmatrix} I & I \\ I & -I \end{bmatrix} \begin{bmatrix} F_{Q/2} & 0 \\ 0 & BF_{Q/2} \end{bmatrix} = \begin{bmatrix} F_{Q/2} & BF_{Q/2} \\ F_{Q/2} & -BF_{Q/2} \end{bmatrix}. \quad (20.16)$$

Figure 20.2 contains the finished quantum circuit.

20.1.2 Period Finding Using the QFT

Now that we've seen how to implement the Quantum Fourier Transform as a quantum circuit, it's time to answer our second question. *What comes out when we measure and how can we use it to learn s ?*

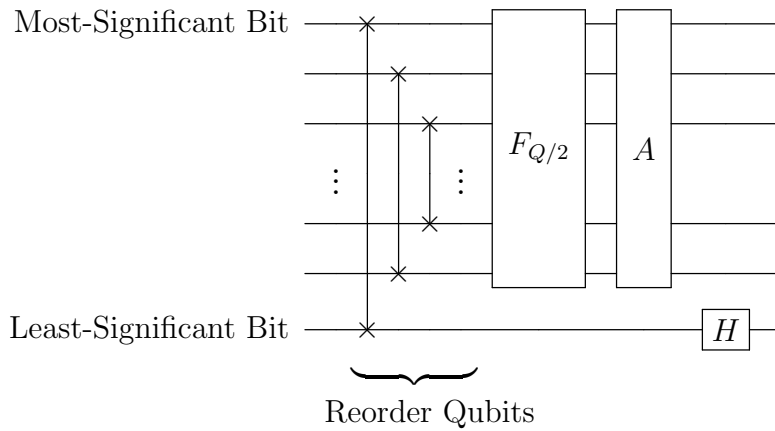


Figure 20.2: Complete circuit for recursively implementing the Quantum Fourier Transform.

Recall from the previous lecture that after we compute $f(r)$ and then measure the second register, we have a quantum state of the form

$$|\psi\rangle = \frac{|r\rangle + |r+s\rangle + |r+2s\rangle + \dots + |r+(L-1)s\rangle}{\sqrt{L}} \tag{20.17}$$

After we apply F_Q , as described in Equation 20.1, the above state is transformed to

$$\frac{1}{\sqrt{QL}} \sum_{k=0}^{Q-1} \sum_{l=0}^{L-1} \omega^{(r+ls)k} |k\rangle. \tag{20.18}$$

What's going on with the above state? Let's start with an easy special case, and only later handle the general case. The easy case is that s divides Q . Assuming that s divides Q , let's answer the question: *which k 's can be observed, when we measure the above state in the computational basis?*

What we want to know is, for a given k , do the various contributions to k 's amplitude interfere constructively or destructively? To answer this question we can ignore the global phase ω^{rk} and just look at the sum

$$\sum_{l=0}^{L-1} \omega^{kls}$$

The key is to identify whether ks is a multiple of Q . If ks is not a multiple of Q then we have *destructive* interference because the terms $\omega^{ks}, \omega^{2ks}, \omega^{3ks},$

etc... are all pointing in different directions in the complex plane and they cancel each other out.

Just like the thumbtack's movement coming back to the origin.

If ks is a multiple of Q , or equivalently, if $k = c\frac{Q}{s}$ for some integer c , then we have *constructive* interference. Since ω was a Q^{th} root of unity the terms ω^{ks} , ω^{ks^2} , ω^{3ks} , etc... all point in the same direction in the complex plane.

If we repeat the algorithm several times then we'll generate a list of such k 's, each of which is an integer multiple of $\frac{Q}{s}$. Afterwards, one can show that we just need to take their GCD to get $\frac{Q}{s}$ itself (with overwhelming probability), from which we can compute s , given our knowledge of Q .

Remember: This is only possible because we assumed that s divides Q .

The harder, general case is that s *doesn't* divide Q . In this case, if we calculate the final amplitude for a specific basis state k , then ignoring the global phase ω^{kr} and the normalization, we'll still get a sum of the form $\sum_{l=0}^{L-1} \omega^{ksl}$. So, how likely we are to observe k will still depend on whether this sum constructively or destructively interferes. What changes is that now Q/s isn't an integer, and as a result, neither the constructive nor the destructive interference will be perfect. But we'll see that they're still good enough for the period s to be efficiently recovered.

If $k \neq \lfloor c\frac{Q}{s} \rfloor$ then we'll claim that we see mostly destructive interference.

We claim that if k is *close* to an integer multiple of $\frac{Q}{s}$, then we mostly see *constructive* interference. If k is *far* from any integer multiple of $\frac{Q}{s}$, then we'll see mostly *destructive* interference.

Let's look at the constructive case first. Assume we get a bit lucky and we have (say) $k = c\frac{Q}{s} + \epsilon$ where $|\epsilon| \sim \frac{1}{10}$. That means that, ignoring normalization, the final amplitude of basis state k has the form

$$\sum_{l=0}^{L-1} \omega^{(c\frac{Q}{s} + \epsilon)sl} = \sum_{l=0}^{L-1} \omega^{cQl} \omega^{\epsilon sl}. \quad (20.19)$$

We can drop the ω^{cQl} because $\omega^{cQl} = e^{(2\pi i/Q)(cQl)} = e^{2\pi icl} = 1$. We're then left with just the $\omega^{\epsilon sl}$ part. For the sake of clarity we can rewrite this term as $\sum_{l=0}^{L-1} e^{(2\pi i\epsilon)\frac{s}{Q}l}$. Recall now that $L \sim \frac{Q}{s}$. As such, the sum above corresponds to a sum over complex numbers constrained to an ϵ fraction of the unit circle,

ranging roughly from 1 to $e^{2\pi i\epsilon}$. Assuming ϵ is relatively small, this means the complex numbers all point in close to the same direction and so we mostly have constructive interference. This situation is illustrated in Figure 20.3.

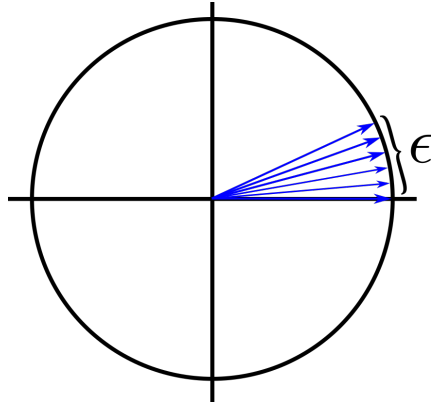


Figure 20.3: Sum over complex numbers constrained to an ϵ fraction of the unit circle. So long as ϵ is relatively small we still have mostly constructive interference.

Next suppose k isn't close to an integer multiple of $\frac{Q}{s}$. In that case, as we vary l , the term ω^{ksl} will loop all the way around the unit circle one or more times. As such, we'll get mostly destructive interference except for a small amount of constructive interference from the final rotation. If you plot the final amplitude as a function of k , you get something like the graph in Figure 20.4.

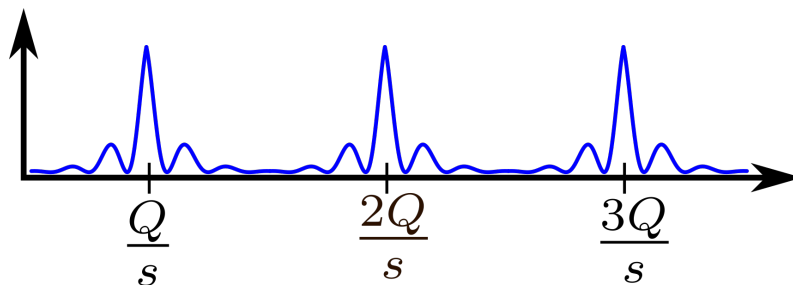


Figure 20.4: Sketch of the magnitude of amplitudes in our post-QFT quantum state Shor's algorithm when s does *not* divide Q .

Lecture 21: Continued Fractions and Shor's Algorithm Wrap-Up

In this lecture we'll finish Shor's algorithm and then discuss some of its implications. Last we saw our protagonists, they were in a superposition of the form

$$|\psi\rangle = \frac{|r\rangle + |r+s\rangle + |r+2s\rangle + \cdots + |r+(L-1)s\rangle}{\sqrt{L}}$$

and we were trying to use the Quantum Fourier Transform (QFT) to extract the period s . Our first order of business was to give a polynomial-size quantum circuit to implement the QFT. Our second order of business was to understand what we observe after we apply the QFT and then measure in the computational basis. Recall that there were two cases:

- ▶ If s divides Q then each possible measurement outcome has either perfectly constructive or perfectly destructive interference. The outcomes with constructive interference are all integer multiples of $\frac{Q}{s}$. From a few random outcomes it's easy to recover s itself, given our knowledge of Q .
- ▶ If s doesn't divide Q then the pattern of constructive and destructive interference is no longer perfect, but has some noise to it. That's because $c\frac{Q}{s}$ is no longer generally an integer, but the algorithm's output still needs to be an integer. So we effectively get a rounding effect, where the nearest integers to $c\frac{Q}{s}$ have the strongest constructive interference.

We've addressed the first case, so we'll now focus on the second case. Say we run the algorithm once getting an integer $k_1 = \lfloor c_1 \frac{Q}{s} \rfloor$ and then run it again to get k_2, k_3 , etc. The question is then, given these integers almost all of which are close to integer multiples of $\frac{Q}{s}$, *how do we use them to deduce s itself?*

21.1 Continued Fraction Algorithm

This brings us to the final step of Shor's algorithm, which is another piece of classical number theory called the **Continued Fraction Algorithm**. Whenever an outcome k is observed, we'd like to determine whether it's close to an integer multiple of $\frac{Q}{s}$ and if so what the multiple is; this is where we use continued fractions. Continued fractions are those expressions like

$$1 + \frac{1}{1 + \frac{1}{1 + \dots}} = \phi = \frac{1 + \sqrt{5}}{2}.$$

How do continued fractions help us approximate rational numbers? It's easiest to illustrate with an example, so let's look at the continued fraction expansion of an approximation of π , 3.14.

$$3.14 = 3 + \frac{14}{100} = 3 + \frac{1}{\frac{100}{14}} = 3 + \frac{1}{7 + \frac{2}{14}} = \dots$$

The idea is that we keep pulling out the largest integer we can and rewriting our expression until we have an approximation of $\frac{Q}{s}$ to within an accuracy of about $\frac{1}{Q^2}$. The reason why the method works is that s is a relatively small integer, so $\frac{Q}{s}$ is not only rational but has a relatively small denominator. In more detail, let's write

$$k = c \frac{Q}{s} \pm \epsilon \tag{21.1}$$

where ϵ is some small value. Then we divide the above equation through by Q to get

$$\frac{k}{Q} = \frac{c}{s} \pm \frac{\epsilon}{Q} \tag{21.2}$$

This immediately implies the following inequality

$$\left| \frac{k}{Q} - \frac{c}{s} \right| \leq \frac{\epsilon}{Q} \tag{21.3}$$

We'll exploit the key inequality above, along with the following:

- ▶ We know l .
- ▶ We know Q (because we picked it, it's some power of 2 of order N^2).

- We know that s isn't too large.

$s \leq N$ because the order of the multiplicative group is less than N , and the order of any element in the group is at most the number of elements.

So $\frac{k}{Q}$ isn't just close to any rational number $\frac{c}{s}$, it's close to a rational number with a pretty small denominator, and that doesn't happen by random chance. This is the reason why we set Q to be $\sim N^2$ in the first place; doing so ensures that the rational approximation $\frac{c}{s}$ to $\frac{k}{Q}$ is more-or-less unique and moreover that there's an efficient algorithm to find it.

There's math that backs this up, we're just not covering it here.

Suppose I give you a rational number, say 0.25001, and I tell you that it's close to a rational number with an unusually small denominator. *How could you figure out which such rational number it's close to without having to try all possible small denominators, of which there might still be too many?* In this particular example you just stare at the thing and immediately see that $\frac{1}{4}$ is the answer! OK, but *what would be a more systematic way of doing it?*

A more systematic way is to expand the input number as a continued fraction until the leftover part is so small that we can safely discard it. To illustrate:

$$.25001 = \frac{25001}{100000} = \frac{1}{\frac{100000}{25001}} = \frac{1}{3 + \frac{24997}{25001}} = \frac{1}{3 + \frac{1}{\frac{25001}{24997}}} = \frac{1}{3 + \frac{1}{1 + \frac{4}{24997}}}$$

Now we've reached $\frac{4}{24997}$, a number small enough for us to discard, which leaves us with

$$\frac{1}{3 + \frac{1}{1 + \frac{4}{24997}}} \approx \frac{1}{3 + \frac{1}{1}} = \frac{1}{4}$$

So now we have a way to find $\frac{c}{s}$. *Are we done?* Well, we still have the same difficulty that we encountered in the s divides Q case. Namely that k and s might share a nontrivial divisor. If, for example, k and s were even, then we'd have no possible way to tell $\frac{c}{s}$ apart from $\frac{c/2}{s/2}$. We solve this using exactly the same approach as before. We repeat the algorithm several times to generate

$\frac{c_1}{s_1}, \frac{c_2}{s_2}, \frac{c_3}{s_3}$, etc. . . One can then show that the least common multiple of the s_i 's will be s itself, with high probability.

*Today, half of pop-science articles **still** say that “quantum computers would factor numbers by trying all the possible divisors in parallel.” If you’ve taken anything away from our discussion of how Shor’s algorithm works, I hope you now agree that it’s more subtle than that! In the next lecture, we’ll see how a quantum speedup for “pure, brute-force search” does exist, but it’s not exponential, but “merely” quadratic.*

21.2 Applications of Shor’s Algorithm

The ink wasn’t dry on Shor’s paper before people started asking: *what else might Shor’s algorithm be good for, besides factoring?*

For starters, as we mentioned a couple lectures ago and as Shor showed in his original paper, it also gives exponential speedup for **Discrete Log**, which is the following problem: given a prime p , and integers g and a , find an x such that $g^x = a \pmod p$. This is how Shor’s algorithm breaks the Diffie-Hellman cryptosystem.

It was noted shortly afterward that Shor’s algorithm can also be modified to break Elliptic Curve cryptosystems. Indeed, people quickly figured out that Shor’s algorithm can be modified to solve pretty much any problem related to finding hidden structures in abelian groups. Almost all the public-key cryptosystems that we currently use in practice involve finding such hidden structures.

In the years after Shor’s algorithm, a lot of research in quantum algorithms was directed towards answering the question of *to what extent can we generalize Shor’s algorithm to solve problems about non-abelian groups?* By now, though, many people have given up on this research direction. It turns out that finding hidden structures in non-abelian groups is very, very hard.

Why did people care about non-abelian groups? Well, if Shor’s algorithm could be generalized to handle them, there are two famous problems that would help us solve.

21.2.1 Graph Isomorphism

In the **Graph Isomorphism** problem, we're given two undirected graphs and need to decide whether they're isomorphic—that is, whether there's some permutation of vertex labels such that the two graphs have the same edges. Graph isomorphism is a problem that no one yet knows how to solve in polynomial time, but that famously seems to have “too much structure” to be **NP**-complete. In the early 1970s when Leonid Levin co-discovered the theory of **NP**-completeness legend has it that he sat on his discovery for more than a year because he was trying to show that Graph Isomorphism is **NP**-complete—something that we now believe is impossible.

People quickly realized that if you could generalize Simon's and Shor's algorithms to a situation where the underlying group is the symmetric group \mathcal{S}_n , instead of an abelian group like \mathbb{Z}_2^n or \mathbb{Z}_N^\times , then it would be possible to solve Graph Isomorphism in quantum polynomial time. In 2016, though, Babai (who's been studying Graph Isomorphism for forty-plus years) found a classical algorithm to solve Graph Isomorphism in quasipolynomial time, meaning $\mathcal{O}(n^{\text{polylog}(n)})$. Many people suspect that Graph Isomorphism is in **P**, for one thing because the problem is easy in practice almost all of the time. In any case, since we now know that Graph Isomorphism is at worst quasipolynomial classically, there's no longer any possibility of getting an exponential quantum speedup for the problem.

21.2.2 Lattice-Based Cryptography

There's a type of public-key cryptography called **Lattice-Based Cryptography**, which is becoming increasingly important theoretically and even practically, and which we don't know how to break (yet) even with a quantum computer. Given a collection $\{z_0, \dots, z_{n-1}\}$ of vectors in \mathbb{R}^n , the *lattice* spanned by the collection is the set of all integer linear combinations of the vectors:

$$L = \{a_0 z_0 + \dots + a_{n-1} z_{n-1} \mid a_0, \dots, a_{n-1} \in \mathbb{Z}\} \quad (21.4)$$

A typical problem relevant to lattice-based cryptography would be, for example: given $\{z_0, \dots, z_{n-1}\}$, find the shortest nonzero vector in L —or at least, a vector that's within a \sqrt{n} factor of being the shortest. It turns out that you can create entire public-key cryptosystems around these sorts of problems.

There was an important result by Oded Regev in 2005 which says that we could break lattice-based cryptography if we could generalize Shor's algorithm to work for a nonabelian group called the dihedral group. Needless to say (because otherwise I would've told you!), no one has yet succeeded in

doing so. So, lattice-based cryptography is currently an attractive alternative to RSA and Diffie-Hellman for those who are paranoid about quantum computers. But it's also attractive for other reasons, including the prospect of **Fully Homomorphic Encryption**: the ability to do arbitrary computations on encrypted data without ever decrypting it. This would let people submit their data to cloud computing servers and then get back the results without the cloud server ever learning what computation it did. In 2009 Craig Gentry proposed the first fully homomorphic encryption scheme using lattice-based crypto; since then other schemes have been proposed. Again, these are not encryption schemes that we know how to break even using a quantum computer. There's still a practical problem with these schemes: the key sizes, message sizes, and computation times tend to be large. But the schemes have been steadily improving, and many of them are now either practical or nearing practicality.

Lecture 22: Grover's Algorithm

The next quantum algorithm we'll cover is **Grover's Algorithm** which was discovered in 1995 shortly after Shor's algorithm.

Both Grover and Shor were working at Bell Labs at the time.

Grover's algorithm gives a smaller speedup than Shor's (quadratic rather than exponential), but for a much wider range of problems. Just like with the other quantum algorithms we've seen, it's easiest to think of Grover's algorithm in the black box setting. Given an oracle function $f : \{0, \dots, N-1\} \rightarrow \{0, 1\}$ we'd like to answer two questions:

- ▶ Is there an x such that $f(x) = 1$?
- ▶ If so, what is such an x ?

The basic problem that Grover's algorithm addresses is unordered search, that is looking through an unstructured list of bits for a 1 bit. Classically we'd need a linear number of queries, $\Omega(N)$, to solve this problem deterministically. *Why?* Simply, because if we want to know for certain whether there's a treasure hidden in one of N boxes, then even after opening $N-1$ boxes and finding them empty we still need to open the N^{th} box! Even if we only care about how long it takes on average, and we know a treasure is guaranteed to be in *some* box, finding it takes still takes $\sim \frac{N}{2}$ queries, which is linear in N . Grover's algorithm solves both problems, with high probability, using only $\mathcal{O}(\sqrt{N})$ quantum queries to the function f .

This might sound impossible—but, as we'll see, it's quite similar to how the Elitzur-Vaidman bomb worked.

The number of qubits needed to run Grover's algorithm is very low, $\mathcal{O}(\log(N))$, and the number of gates required (besides those needed to compute f itself)

is also reasonable, $\mathcal{O}(\sqrt{N} \log(N))$. However, for Grover's algorithm to work we do need to assume that we have access to f that lets us apply the unitary transformation $|x, a\rangle \rightarrow |x, a \oplus f(x)\rangle$. This wasn't important in Shor's Algorithm because we only made one query and then discarded the result.

There are two main example applications to keep in mind with Grover's algorithm. The first application is solving combinatorial search and optimization problems, such as **NP**-complete problems. Here, we think of $N = 2^n$ as being exponentially large, and we think of each candidate solution $x \in \{0, 1\}^n$ as an n -bit string. We then set, for example, $f(x) = \varphi(x)$, where φ is an instance of Satisfiability or some other **NP**-complete problem. Then, Grover's algorithm can find an x such that $\varphi(x) = 1$ in $\mathcal{O}(2^{n/2})\text{poly}(n)$ time. That is, $N = 2^{n/2}$ queries to f , and $\text{poly}(n)$ time to implement each query (say, by checking whether a given x satisfies φ). This is an apparent speedup for **NP**-complete problems—but at most a quadratic one and also only conjectural, because of course we can't even rule out the possibility of **P=NP**, which would annihilate this sort of speedup.

For an **NP**-complete problem like CircuitSAT, we can be pretty confident that the Grover speedup is real, because no one has found any classical algorithm that's even slightly better than brute force. On the other hand, for more “structured” **NP**-complete problems, we do know exponential-time algorithms that are faster than brute force. For example, 3SAT is solvable classically in about $\mathcal{O}(1.3^n)$ time. So then, the question becomes a subtle one of whether Grover's algorithm can be combined with the best classical tricks that we know to achieve a polynomial speedup even compared to a classical algorithm that uses the same tricks. For many **NP**-complete problems the answer seems to be yes, but it need not be yes for all of them.

The second example application of Grover's algorithm to keep in mind is searching an actual physical database. Say you have a database of personnel records and you want to find a person who matches various conditions (hair color, hometown, etc.). You can set $f(x) = 1$ if person x meets the criteria and $f(x) = 0$ otherwise. Grover's algorithm can search for an x such that $f(x) = 1$ in $\mathcal{O}(\sqrt{N})$ steps. One big advantage of Grover's algorithm as applied to actual physical databases is that the quantum speedup is provable; it doesn't rely on any unproved computational hardness assumptions.

Some people have questioned the practicality of using Grover's algorithm to search a physical database, because the database needs to support “superposed queries.” That is, you need to be able to query many records in superposition and get back a superposition of answers. A memory that would support these kinds of queries is called a “quantum RAM.” Building one is a whole additional technological problem beyond building a quantum computer itself. It remains

unclear whether people will be able to build quantum RAMs without n active, parallel computing elements—which, if you had them, would remove the need to run Grover's algorithm.

In this lecture, though, we'll treat such things as “mere engineering difficulties”!

22.1 The Algorithm

OK, so without further ado, *how does Grover's algorithm work?* For simplicity, let's assume that a solution exists and is unique. We call the unique x^* such that $f(x^*) = 1$ the “marked item”. We'll also assume for simplicity that $N = 2^n$ is a power of 2. This will allow us to do our favorite trick: start by Hadamarding n qubits. Doing so brings the initially all-0 state to a uniform superposition

$$\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle, \quad (22.1)$$

where each $x \in \{0, \dots, N-1\}$ is represented as an n -bit string. Then we query with U_f , a unitary transformation that flips the amplitude of the marked item:

$$U_f |x\rangle = (-1)^{f(x)} |x\rangle. \quad (22.2)$$

As we've already seen in this course, if we can apply $|x, a\rangle \rightarrow |x, a \oplus f(x)\rangle$, then we can also apply the phase oracle $|x\rangle \rightarrow (-1)^{f(x)} |x\rangle$. Phase oracles are more convenient for the purposes of Grover's algorithm.

Next, we apply a unitary matrix D below.

$$D = \begin{bmatrix} \frac{2}{N} - 1 & \frac{2}{N} & \cdots & \frac{2}{N} \\ \frac{2}{N} & \frac{2}{N} - 1 & \cdots & \frac{2}{N} \\ \vdots & \cdots & \ddots & \vdots \\ \frac{2}{N} & \cdots & \cdots & \frac{2}{N} - 1 \end{bmatrix} \quad (22.3)$$

This is the so-called **Grover Diffusion Operator**, which has the effect of

flipping all N amplitudes about the mean amplitude $\bar{\alpha} = \frac{1}{N} \sum_{x=0}^{N-1} \alpha_x$,

$$\alpha_x \rightarrow 2\bar{\alpha} - \alpha_x. \quad (22.4)$$

So why does applying D help us? Well, let's look at what's happening after a single Grover iteration of applying U_f and D pictorially, using the depiction shown in Figures 22.1–22.4. After a single diffusion operation, we've managed to increase the amplitude of the marked item to roughly $\frac{3}{\sqrt{N}}$ and decrease the amplitudes of all the other items accordingly.

Then, we keep repeating by applying another U_f and then another D and so on. By doing so, we can increase the amplitude of the marked item further as pictured in Figures 22.5–22.7.

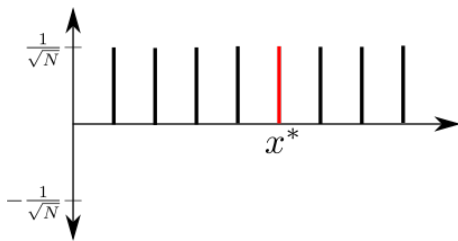


Figure 22.1: The initial amplitudes of the system, an even superposition state.

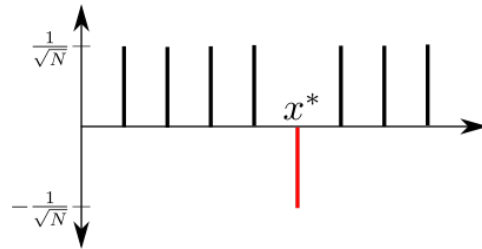


Figure 22.2: The amplitudes following the first application of the phase oracle. Note that the amplitude of the marked item has had its sign flipped.

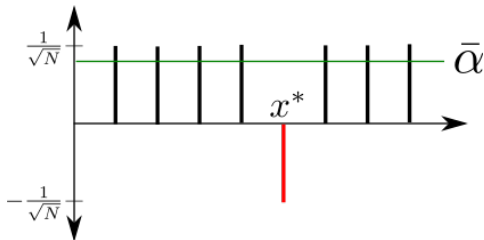


Figure 22.3: The average amplitude $\bar{\alpha}$ has been explicitly drawn in.

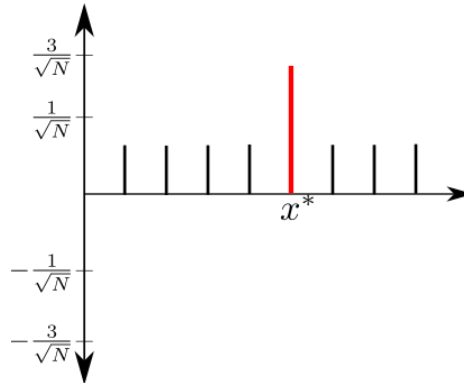


Figure 22.4: The amplitudes following the first Grover diffusion operator.

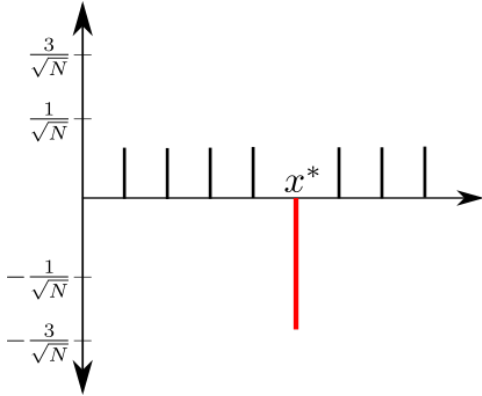


Figure 22.5: Amplitudes following the second application of the phase oracle. Note that the amplitude of the marked item has had its sign flipped again.

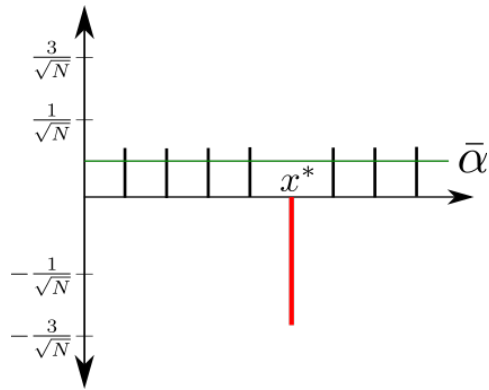


Figure 22.6: Amplitudes following the second application of the phase oracle with the new average amplitude $\bar{\alpha}$ explicitly drawn in.

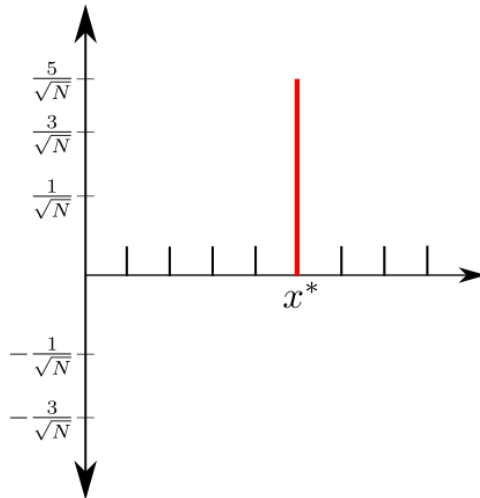


Figure 22.7: Amplitudes following the second Grover diffusion operator.

As an approximation, we can say that when the number of queries is small, the repetition increases the marked item's amplitude as $\frac{1}{\sqrt{N}}, \frac{3}{\sqrt{N}}, \frac{5}{\sqrt{N}}, \frac{7}{\sqrt{N}} \dots$. Notice that it would take $\mathcal{O}(\sqrt{N})$ steps for this series to reach $\frac{\sqrt{N}}{\sqrt{N}} = 1$. This represents a quadratic speedup: classically, we'd need about $\frac{t}{N}$ queries to find the answer, since after t queries we'd have a probability of $\frac{t}{N}$ of having found the marked item. Meanwhile, if we measure after t queries in Grover's algorithm we find the marked item with probability of order $\sim \left(\frac{2t}{\sqrt{N}}\right)^2 = \frac{4t^2}{N}$. This picture isn't exactly right, though, because we ignored some details. Over time the mean gets smaller, so the increase in the marked item's amplitude slows down.

Which makes sense, because otherwise the amplitude would continue increasing past 1! We'll see exactly what happens shortly.

First, though, a natural question to ask about Grover's algorithm is *why should it take \sqrt{N} steps? Why not $\sqrt[3]{N}$ or $\log N$?* We see here that in some sense the ultimate source of the N is the fact that amplitudes are the square roots of probabilities. Instead of adding $\sim \frac{1}{N}$ probability to the marked item with each query, quantum mechanics lets us add $\sim \frac{1}{\sqrt{N}}$ amplitude, resulting in quadratically faster convergence. This intuition will be made more rigorous in the next lecture when we learn about the BBBV Theorem.

22.1.1 Implementing the Diffusion Operator

Of course, if we want to use Grover's algorithm in practice, then in addition to bounding the number of queries by $\mathcal{O}(\sqrt{N})$, we'll also need to find a small quantum circuit to implement the Grover diffusion operator D . Say we want to implement D on an n -qubit state ($N = 2^n$). It's easiest if we look at what D does in the Hadamard basis. In the Hadamard basis, the y^{th} amplitude would be

$$\beta_y = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} (-1)^{y \cdot x} \alpha_x. \quad (22.5)$$

The first of these amplitudes plays a special role. If $y = 0$ we have $\frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} \alpha_x$ which is proportional to the average, which is good because our goal was to invert about the average. The other y values play no particular role in Grover's algorithm.

So, in the Hadamard basis what we want is to perform the diagonal matrix A ,

$$A = \begin{bmatrix} 1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{bmatrix}. \quad (22.6)$$

This A is easy to implement as a quantum circuit by using some ancilla qubits to check whether the input is all 0's, inverting the phase if not, and finally uncomputing garbage (the details of this implementation are left as an exercise). In order to implement D we therefore:

1. Move to the Hadamard basis using a round of Hadamard gates applied to each of the qubits.
2. Apply A .
3. Move back to the computational basis by applying another round of Hadamards.

The final circuit for implementing Grover's algorithm is shown in Figure 22.8.

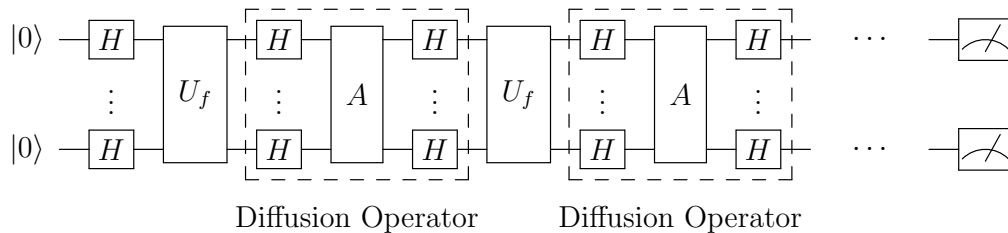


Figure 22.8: Complete circuit for running Grover's algorithm.

22.1.2 Geometric Interpretation

We described above how the Grover diffusion operator D in Equation 22.3 has the effect of flipping all of the amplitudes about the mean. It will be useful to describe a different way to think about the Grover diffusion operator's effect.

The first thing to note is that we can write the unitary operation A in Equation 22.6 as

$$A = \begin{bmatrix} 1 & & & \\ & -1 & & \\ & & \ddots & \\ & & & -1 \end{bmatrix} = 2|0\rangle\langle 0| - I. \quad (22.7)$$

When applied to a general state,

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle + \cdots + \alpha_{N-1}|N-1\rangle,$$

this produces

$$A|\psi\rangle = -\alpha_0|0\rangle + \alpha_1|1\rangle \cdots \alpha_{N-1}|N-1\rangle.$$

In other words, the A matrix flips the sign of all of the amplitudes except that of the $|0\rangle$ state (equivalently you could say A flips the sign of α_0 while leaving the rest of the amplitudes alone). This has the effect of reflecting our state about the $|0\rangle$ axis in the N -dimensional Hilbert space of the system.

Recall that $D = H^{\otimes n}AH^{\otimes n}$. Thus,

$$\begin{aligned} D &= H^{\otimes n}AH^{\otimes n} \\ &= H^{\otimes n}(2|0\rangle\langle 0| - I)H^{\otimes n} \\ &= 2H^{\otimes n}|0\rangle\langle 0|H^{\otimes n} - H^{\otimes n}IH^{\otimes n} \\ &= 2|\phi\rangle\langle\phi| - I, \end{aligned} \quad (22.8)$$

where $|\phi\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle$ is the uniform superposition state. Notice the similarity between the form of this operator and the form of A in Equation 22.7. Indeed, just as we saw that the result of applying A is a reflection of the state about the $|0\rangle$ axis in N -dimensional Hilbert space, D reflects about the $|\phi\rangle$ axis.

Similarly, U_f corresponds to a reflection about the $|x^*\rangle$ axis in N -dimensional Hilbert space, where $|x^*\rangle$ is the basis state corresponding to the marked item.

22.1.3 Analysis

Now let's analyze Grover's algorithm more carefully and actually prove that it works.

The initial state of the system following the first round of Hadamards shown in Figure 22.8 is

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle. \quad (22.9)$$

Somewhere in the N -dimensional space is the basis state $|x^*\rangle$ corresponding to the marked item we're looking for. Our initial state $|\psi\rangle$ overlaps only very slightly with the state of the marked item $|x^*\rangle$: $\langle\psi|x^*\rangle = \frac{1}{\sqrt{N}}$. So $|\psi\rangle$ and $|x^*\rangle$ are not quite orthogonal, but nearly so. Now, these two states $|\psi\rangle$ and $|x^*\rangle$ span a two-dimensional subspace of the overall N -dimensional Hilbert space. A crucial insight about Grover's algorithm is that it operates entirely within this subspace. *Why?* Simply because if we start in the subspace then neither the queries nor the Grover diffusion operations ever cause us to leave it! This means that we can visualize everything Grover's algorithm is doing by just drawing a picture in the 2D plane. The axes here are given by $|x^*\rangle$ and $|\text{unmarked}\rangle$, where $|\text{unmarked}\rangle$ is an equal superposition state over all of the unmarked items:

$$|\text{unmarked}\rangle = \frac{1}{\sqrt{N-1}} \sum_{x \neq x^*} |x\rangle. \quad (22.10)$$

Note that $|x^*\rangle$ and $|\text{unmarked}\rangle$ are clearly orthogonal to each other. We've seen already how the algorithm alternates between two types of operations:

- Inverting the component of our state that points in the $|x^*\rangle$ direction by querying U_f as shown in Figure 22.9.

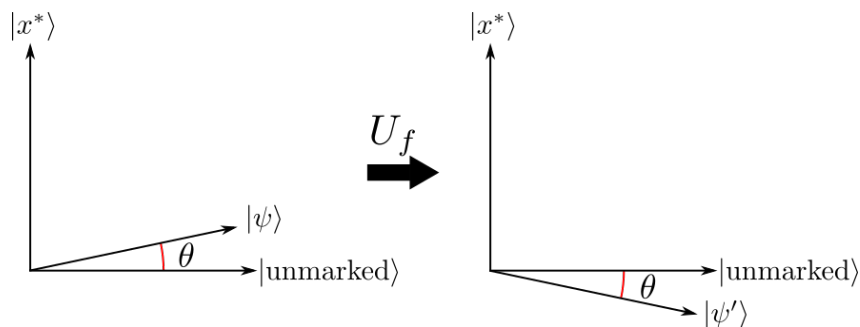


Figure 22.9: Applying the oracle U_f to the initial state $|\psi\rangle$ to get the new state $|\psi'\rangle$.

- Reflecting our state about $|\phi\rangle$, the uniform superposition state, using the diffusion operator D , as shown in Figure 22.10:

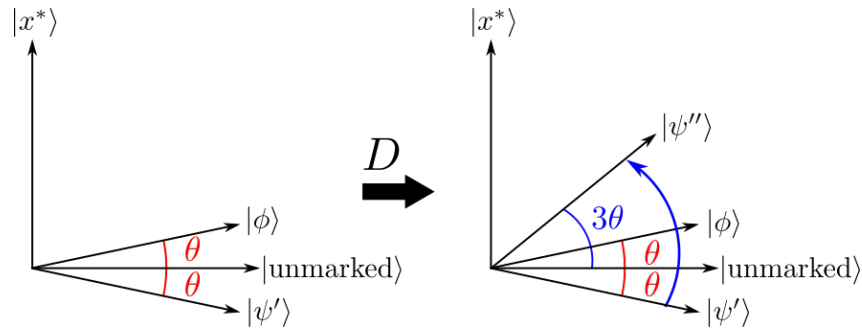


Figure 22.10: Applying the Grover diffusion operator D to reflect $|\psi'\rangle$ about the state $|\phi\rangle$ and get the new state $|\psi''\rangle$.

Initially, the angle of $|\psi\rangle$ with the horizontal is $\theta = \arcsin\left(\frac{1}{\sqrt{N}}\right) \approx \frac{1}{\sqrt{N}}$. After each iteration, we've rotated by an additional $\frac{2}{\sqrt{N}}$. Hence, the probability of success after the t^{th} iteration is given by

$$P(\text{success}) = |\langle x^* | \psi \rangle|^2 = \sin^2\left(\frac{2t+1}{\sqrt{N}}\right). \quad (22.11)$$

This means that we'll get super close to $|x^*\rangle$ and have a high probability of observing x^* if we measure after about $\frac{\pi}{4}\sqrt{N}$ iterations—something that you can directly see from the geometric picture. We might not get *exactly* to 1 if the step size of the rotations causes us to overshoot slightly, but at any rate we'll get close.

We can see right away that, unlike any classical algorithm, Grover's algorithm has the amusing property that its success probability starts getting *worse* if we run it for too long!

Grover's algorithm has been compared to baking a soufflé. Once risen it must be taken out of oven or it'll deflate again. On the other hand Grover's algorithm has at least one property not shared by soufflés. Namely, if you "leave it in the oven" for even longer it rises a second time, then goes down a second time and so on forever!

Graphing the success probability as a function of the number of iterations produces a sinusoidal curve as seen in Figure 22.11. Grover's algorithm can

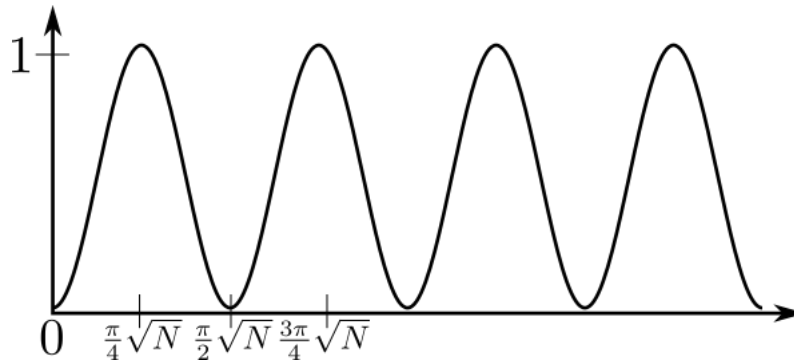


Figure 22.11: Success probability of Grover's algorithm as function of the number of iterations, assuming 1 marked item out of N .

also put you into an interesting dilemma: suppose you've run the algorithm for a given number of iterations, fewer than $\frac{\pi}{4}\sqrt{N}$. Then you could measure right away and take your chances with whether you'll observe the solution, or you could let it run longer to boost your chances. If you measure right now and don't see the solution then you need to start over from the very beginning.

This could make for an interesting science-fiction story: the heroes need to break a cryptographic code to beat the villains, so they run Grover's algorithm over all the possible decryption keys. They've run it for a day and gotten up to 45% probability of observing the solution, but now the villains have entered their compound and are closing in on them. So, do they measure now or do they let the algorithm run a bit more?

Are you better off measuring after fewer than $\frac{\pi}{4}\sqrt{N}$ iterations? It depends on your desired level of confidence in getting the right answer. If your goal is just to minimize the average number of queries until you learn the answer, then it turns out to be optimal to stop after $c\sqrt{N}$ iterations, for some $c < \frac{\pi}{4}$ (details left as an exercise).

22.1.4 Multiple Marked Items

For simplicity, above we assumed exactly one marked item. *What happens if there are more?* The simple answer is that the entire cycle happens faster. In particular, if K items out of N are marked then the success probability for

Grover's algorithm peaks at $\frac{\pi}{4}\sqrt{\frac{N}{K}}$ queries. This is straightforward to see in the geometric picture. With K marked items, our two-dimensional subspace is spanned by the uniform superposition $|\phi\rangle$ and $|x^*\rangle = \frac{|x_1^*\rangle + \dots + |x_k^*\rangle}{\sqrt{K}}$, an equal superposition over all K marked items. These have an inner product

$$\langle\phi|x^*\rangle = \sqrt{\frac{K}{N}}. \quad (22.12)$$

Thus, our state initially starts with an angle of $\theta = \arcsin(\sqrt{\frac{K}{N}}) \approx \sqrt{\frac{K}{N}}$. After each iteration the state rotates an additional 2θ toward the $|x^*\rangle$ axis, meaning that after t iteration the success probability is

$$P(\text{success}) = \sin^2\left((2t+1)\sqrt{\frac{K}{N}}\right) \quad (22.13)$$

Hence, $P(\text{success}) \approx 1$ after about $T = \frac{\pi}{4}\sqrt{\frac{N}{K}}$ iterations.

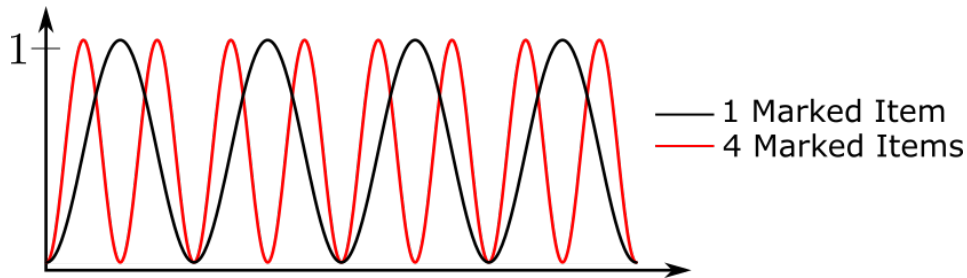


Figure 22.12: Success probability of Grover's algorithm as function of the number of iterations with more than one marked item. Each additional marked item causes the success probability to oscillate with an increased frequency.

One of the first questions people asked about Grover's algorithm was: *what if the number of marked items, K , isn't known?* You can sort of see the danger. It's possible to run Grover's algorithm the right number of times to hit the peak success probability when there's a single marked item, but that might overshoot and lead to success probability near 0 if there are more marked items.

The most basic way to solve this problem is simply to run the algorithm for a random number of iterations, say between 0 and \sqrt{N} . If we do this then most of the time we expect to end up somewhere around the middle of a sinusoid (neither at a trough nor a peak), where we have a constant probability (say, 40% or 60%) of observing a solution if we measure. This is perfectly sufficient

from an algorithmic standpoint, since it means that we only need to repeat the algorithm $\mathcal{O}(1)$ times on average until we see a marked item. This gives us an upper bound of $\mathcal{O}(\sqrt{N})$ queries to find a marked item with high probability, regardless of how many marked items there are (assuming there's at least one).

What happens if we run Grover's algorithm, but the database turns out to have no marked items? When we query f nothing happens. When we apply the diffusion operator nothing happens. So, the state just remains a uniform superposition over all N items for the entire duration of the algorithm. This means that when we measure we just get a random item. We can check that item and see that it isn't marked.

How can we be certain that there are no marked items? This is the question that arises in the decision version of Grover's algorithm. In fact, no matter how many times we run Grover's algorithm, we never become 100% sure that there are no marked items since we could've just gotten unlucky and failed to find the items every time. However, after $\mathcal{O}(1)$ repetitions, the algorithm has as high a probability as we like (say, $> 99.99\%$) of finding a marked item assuming that there's at least one of them. If, after we've run Grover's algorithm a sufficient number of times, we still haven't found a marked item, then we can deduce that there almost certainly weren't any. This again requires only $\mathcal{O}(\sqrt{N})$ queries.

We've said that if there are K marked items then we find one of them in $\mathcal{O}(\sqrt{N})$ queries without knowing K . In fact we can do even better than that and find a marked item in only $\mathcal{O}(\sqrt{\frac{N}{K}})$ queries; again without knowing K . This is the same performance as if we did know K , *so how does this work?* Assume for simplicity that N is a power of 2. First, we guess that almost all items are marked, do a single query and then measure. If we find a marked item, great. If not, we guess that $\frac{N}{2}$ items are marked and run Grover's algorithm with $K = \frac{N}{2}$. If we find a marked item, great. Next we run Grover's algorithm with $K = \frac{N}{4}$, then $K = \frac{N}{8}$ and so on, repeatedly halving our guess for the number of marked items until either we've found a marked item or we've searched unsuccessfully with $K = 1$.

This method wastes some queries on "wrong" values of K . Crucially, because the number of queries is increasing exponentially, the number of wasted queries is only a constant factor greater than the number of queries used in the final iteration; the one that guesses an approximately correct value of K . Details of the analysis are left as an exercise for the reader.

Let's end by mentioning a different way to handle the case of multiple marked items that achieves essentially the same performance using a purely classical trick. Again suppose we have N items, K of which are marked. We

want to reduce this to the case of just a single marked item. *How do we do that?* Simple, we pick $\frac{N}{K}$ items uniformly at random and then run Grover's algorithm on that subset only. The number of marked items that we'll catch in the subset is well approximated by a Poisson distribution, and one can calculate the probability of catching exactly one marked item in our sample as $\sim \frac{1}{e}$. So, we search that subset of $\frac{N}{K}$ items using Grover's algorithm for the single marked item case (which uses $O(\sqrt{\frac{N}{K}})$ queries). If we don't find a marked item we can try again with a new random subset.

Exercise for the reader: Show that, if there are K marked items and we want to find all of them, we can do that using $\mathcal{O}(\sqrt{NK})$ queries.

Lecture 23: The BBBV Theorem and Applications of Grover's Algorithm

23.1 The BBBV Theorem

It's great that we can get a quadratic speedup with Grover's algorithm, but we were able to get an exponential speedup with Shor's algorithm. So, *why can't we get a bigger speedup for unordered search?* By now you should have some intuition for the differences between Shor's algorithm and Grover's algorithm. Shor's algorithm provided an exponential speedup by orchestrating a very "global" phenomenon, involving an interference effect that revealed the period of a black-box periodic function. Grover's algorithm let us turn a little amplitude into a bigger amplitude by adding $\sim \frac{1}{\sqrt{N}}$ with each query. It's faster than classical brute-force search, but still laborious (still needing $\mathcal{O}(\sqrt{N})$ time).

We're still hand-waving the issue though. We haven't ruled out the possibility of a quantum algorithm that beats Grover, solving unordered search in, say, $\sqrt[3]{N}$ or $\log(N)$ queries or whatever. For that we need a key result of from 1994 called **The BBBV Theorem**. This theorem is named for Bennett, Bernstein, Brassard and Vazirani, and proved that Grover's algorithm is indeed asymptotically optimal for the black-box unordered search problem.

*Note that the BBBV Theorem was published in 1994, so it actually predates Grover. Grover's algorithm thus has the rare distinction of being proved to be optimal **before** it was discovered to exist.*

Amusingly, BBBV were trying to prove that there's no magic way to search faster using a quantum computer. They were able to get a lower bound of

$\Omega(\sqrt{N})$, and figured that tightening the bound to $\Omega(N)$ was a technical issue that they could leave for the future—that is, until Grover came along and showed why such a tightening is impossible!

While by now we know many proofs of the BBBV Theorem, the original (and still most self-contained) proof uses what’s called a *Hybrid Argument*. Imagine we’re using an arbitrary quantum algorithm to search for a single marked item in a list of size N . Without loss of generality we can say that the algorithm makes a total of T queries and follows some sequence of operations that looks like

$$U_0 \rightarrow Q_0 \rightarrow U_1 \rightarrow Q_1 \rightarrow U_2 \rightarrow Q_2 \rightarrow \dots \quad (23.1)$$

where each Q_t is a query and each U_t is some unitary transformation (independent of the data in the list). We start by doing a test run of the algorithm on the all-zero list (without a marked item) and see what happens. What we’ll then show is that if we change the value of some item in the list from 0 to 1 (in other words introduce a marked item), the algorithm won’t notice the change very much. This is done by exploiting the fact that unitary transformations are linear and norm-preserving. More concretely, we’ll show that the final state of the algorithm is at most $O(\frac{T}{\sqrt{N}})$ away from what it would have been had we kept the list all-zero. The argument is called “hybrid” because we’ll create hybrid oracles which answer the first m queries as if they’re the all-zero oracle but then switch partway through, and for the last $T - m$ queries answer as they should for a list with a single 1 entry.

While we’ll only consider algorithms that apply unitary transformations, exactly the same proof carries over to algorithms that have intermediate measurements—because we can always model a measurement by a unitary transformation on a larger set of qubits, just like in the Many-Worlds Interpretation.

We’ll let $|\psi_t\rangle$ denote the state immediately following the t^{th} query. Let $x \in \{0, \dots, N - 1\}$ denote an index of a queried list element and suppose we allow the algorithm to use an unlimited number of ancillary qubits for workspace, the state of which will be denoted w . In general, $|\psi_t\rangle$ can be written as a superposition of the list elements x and the possible values of the

workspace qubits w as follows:

$$|\psi_t\rangle = \sum_{x=0}^{N-1} \sum_w \alpha_{x,w,t} |x, w\rangle. \quad (23.2)$$

Here $\alpha_{x,w,t}$ is the amplitude of the basis state $|x, w\rangle$ following the t^{th} iteration.

In Grover's algorithm there's hardly any workspace to speak of. We do use ancillary qubits to implement the diffusion operator, but those qubits are reset to their original state by the time the diffusion operator is finished. The BBBV Theorem, to be fully general, will allow for unlimited workspace, and still will show that the algorithm needs $\sim \sqrt{N}$ queries.

Before we move on it will be useful to introduce a new quantity called the **Query Magnitude**. The query magnitude for some list element $x \in \{0, 1, \dots, N-1\}$ is given by

$$M_x = \sum_{t=0}^{T-1} \sum_w |\alpha_{x,w,t}|^2. \quad (23.3)$$

Note that $|\alpha_{x,w,t}|^2$ is the probability of finding the item x if we were to measure after the t^{th} iteration. The query magnitude M_x is therefore the sum over all of the probability that we would find item x if we measured at iteration t . Rearranging, we find that the sum of all the query magnitudes is

$$\sum_{x=0}^{N-1} \sum_{t=1}^T \sum_w |\alpha_{x,w,t}|^2 = \sum_{t=1}^T \sum_{x=0}^{N-1} \sum_w |\alpha_{x,w,t}|^2 = \sum_{t=1}^T 1 = T. \quad (23.4)$$

From this it immediately follows that the average query magnitude is

$$\frac{1}{N} \sum_{x=0}^{N-1} M_x = \frac{T}{N}. \quad (23.5)$$

Furthermore, given any list of numbers there must be at least one number in the list whose value is at most the average. Thus let's fix some index $x^* \in \{0, \dots, N-1\}$ such that $M_{x^*} \leq \frac{T}{N}$.

This is sometimes referred to as the "Lake Wobegon Principle," after the fictional town where everyone was above average.

Now that we've defined the query magnitude we can explain the hybrid argument. Picture a table like the one to the right where each row is our database at a particular point in time, with time increasing upwards. Initially the table is filled with zeros, meaning that the oracle answers all queries with zero.

Now we're going to change the table for item x^* , the oracle returns 1 for the last query, and *only* for the last query. The modified table is given in Figure 23.1.

This means that the state of the algorithm after the final query $|\psi_T\rangle$, is going to change from what it was before. Let the new state following the final query be $|\psi'_T\rangle$, where the superscript denotes the fact that we've changed just the final query to return $f(x^*) = 1$.

Figure 23.1: Modified table with the final query changed to return 1 for x^* .

So how much can changing the result of the last query change the final state? Before the final query, the states are identical: $|\psi_t\rangle = |\psi'_t\rangle$ for $t < T$. The effect of the final query is to flip the phase of the amplitudes associated

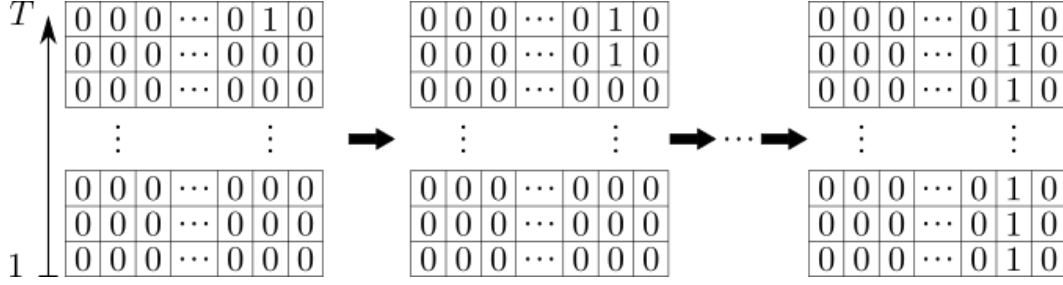


Figure 23.2: For the hybrid argument we repeat the process of swapping out the all-zero oracle for more and more queries until we've replaced it with the oracle returning 1 for x^* for every iteration of the algorithm.

with x^* . In terms of Euclidean distance,

$$\begin{aligned}
 \left\| |\psi_T\rangle - |\psi'_T\rangle \right\| &= \\
 \left\| \left(\sum_{x=0}^{N-1} \sum_w \alpha_{x,w,T} |x, w\rangle \right) - \left(\sum_{x \neq x^*} \sum_w \alpha_{x,w,T} |x, w\rangle - \sum_w \alpha_{x^*,w,T} |x^*, w\rangle \right) \right\| &= \\
 \left\| 2 \sum_w \alpha_{x^*,w,T} |x^*, w\rangle \right\| &= \\
 2 \sqrt{\sum_w |\alpha_{x^*,w,T}|^2} &=
 \end{aligned} \tag{23.6}$$

Suppose now that the hybrid oracle returns 1 for x^* on the last *two* queries. *How much will the output state differ from that in the all-zero case now?* We again know that up to the second-to-last query the states are the same. The distance between $|\psi_{T-1}\rangle$ and $|\psi'_{T-1}\rangle$ is then (following the exact same process as above)

$$\begin{aligned}
 \left\| |\psi_{T-1}\rangle - |\psi'_{T-1}\rangle \right\| &= \left\| 2 \sum_w \alpha_{x^*,w,T-1} |x^*, w\rangle \right\| \\
 &= 2 \sqrt{\sum_w |\alpha_{x^*,w,T-1}|^2}
 \end{aligned} \tag{23.7}$$

But we're not done yet, because what about the final query? We'll define the new vectors $|e_T\rangle = |\psi_T\rangle - |\psi'_T\rangle$ and $|e_{T-1}\rangle = |\psi_{T-1}\rangle - |\psi'_{T-1}\rangle$. Rearranging, we can rewrite $|\psi'_T\rangle$ as $|\psi_T\rangle - |e_T\rangle$ and $|\psi'_{T-1}\rangle$ as $|\psi_{T-1}\rangle - |e_{T-1}\rangle$. Then, by

linearity, the final state is given by

$$Q_T U_T |\psi'_{T-1}\rangle = \underbrace{Q_T U_T |\psi_{T-1}\rangle}_{|\psi'_T\rangle} - Q_T U_T |e_{T-1}\rangle. \quad (23.8)$$

The first term above is precisely the output state $|\psi'_T\rangle$ that we had when we changed only the final query to include the marked item. So we can rewrite the above as

$$Q_T U_T |\psi'_{T-1}\rangle = |\psi_T\rangle - |e_T\rangle - Q_T U_T |e_{T-1}\rangle \quad (23.9)$$

So letting $|\psi''_T\rangle = Q_T U_T |\psi'_{T-1}\rangle$, we have

$$\begin{aligned} \left\| |\psi''_T\rangle - |\psi_T\rangle \right\| &= \left\| |e_T\rangle + Q_T U_T |e_{T-1}\rangle \right\| \\ &\leq \left\| |e_T\rangle \right\| + \left\| Q_T U_T |e_{T-1}\rangle \right\| \\ &= \left\| |e_T\rangle \right\| + \left\| |e_{T-1}\rangle \right\| \\ &= 2 \sqrt{\sum_w |\alpha_{x^*,w,T}|^2} + 2 \sqrt{\sum_w |\alpha_{x^*,w,T-1}|^2} \end{aligned} \quad (23.10)$$

Here the third line crucially used the fact that Q_T and U_T are unitary.

It is straightforward to generalize this to replacing more and more of the queries with ones that return $f(x^*) = 1$, as shown in Figure 23.2. Let's call the state corresponding to replacing the oracle for all T iterations $|\psi_T^{(T)}\rangle$ (where the superscript denotes the number of times we've modified the oracle). Let $M_{x,t} = \sum_w |\alpha_{x,w,t}|^2$, so that $M_x = \sum_{t=1}^T M_{x,t}$, where M_x is the query magnitude of x . Then

$$\left\| |\psi_T\rangle - |\psi_T^{(T)}\rangle \right\| \leq 2 \sum_{t=1}^T \sqrt{\sum_w |\alpha_{x^*,w,t}|^2} = 2 \sum_{t=1}^T \sqrt{M_{x^*,t}}. \quad (23.11)$$

Now recall that, using the ‘‘Lake Wobegon Principle,’’ we had

$$M_{x^*} \leq \frac{T}{N}. \quad (23.12)$$

Subject to the above constraint, the Cauchy-Schwarz inequality tells us that the way to maximize $\sum_{t=1}^T \sqrt{M_{x^*,t}}$ is to set $M_{x^*,1} = \cdots = M_{x^*,T} = \frac{1}{N}$. In that case,

$$\left\| |\psi_T\rangle - |\psi_T^{(T)}\rangle \right\| = 2 \sum_{t=1}^T \sqrt{M_{x^*,t}} \leq \frac{2T}{\sqrt{N}}. \quad (23.13)$$

Finally, in order to achieve a $\Omega(1)$ probability of seeing the marked item x^* when we measure the state at the end of the algorithm, we need the above distance to be $\Omega(1)$ as well. Achieving this requires us to use at least $T \sim \sqrt{N}$ iterations. With that we've proved the BBBV theorem, that any black-box quantum search algorithm requires $\Omega(\sqrt{N})$ queries in order to achieve a constant success probability of seeing the marked item.

The BBBV theorem has since been enormously generalized and now constitutes a whole theory about lower bounds on quantum query complexity. Unfortunately we won't really enter into that in this course, but see Prof. Aaronson's graduate course for more!

23.2 Applications of Grover's Algorithm

23.2.1 OR of ANDs

We can apply Grover's algorithm solve many, many problems that are not quite as simple as unordered search. Our first example of this is the OR of ANDs problem. Suppose we have N input bits arranged into a square table of size $\sqrt{N} \times \sqrt{N}$. The problem is to determine whether there are there any rows with all 1s. Classically, it's clear that in the worst case you have to look through almost the entire table, searching each row until you've found a 0, until you find a row is all 1's.

	0	1	0	1	0	0
	0	0	0	1	1	0
	0	0	0	0	1	0
\sqrt{N}	0	1	1	1	1	0
	1	1	1	1	1	1
	0	0	0	0	1	0
				\sqrt{N}		

Such problems let us encode many other problems that involve multiple quantifiers. For example, in chess we may want to know "Is there a move I can make such that my opponent has no possible response that checkmates me?"

Quantumly we could speed this up by searching each row for a 0 using Grover's algorithm. The query complexity for each row would be $\sim \sqrt{\sqrt{N}} = N^{1/4}$, or technically $N^{1/4} \log(N)$ if we repeat the Grover search on each row enough times to have (say) a $\frac{1}{N}$ probability of error. This means that searching the whole table would take $\sim \sqrt{N} N^{1/4} \log(N) = N^{3/4} \log(N)$ queries.

Alternatively, we could do Grover's algorithm over all the rows, counting each row as a "marked item" if and only if a classical algorithm (which we run as an inner loop) finds no zero in that row. This also has an $\sim N^{3/4} \log(N)$ runtime.

Naturally the next idea is to run Grover's algorithm recursively, where the outer Grover (over the rows) will count a given row as being marked if and only if the inner Grover failed to find a zero in that row. Again, because Grover's algorithm has some probability of error we have to repeat the inner runs about $\log(N)$ times to push the error probability per row down to about $\frac{1}{N}$. So our final query complexity is

$$\mathcal{O}\left(\underbrace{N^{1/4}}_{\text{Outer Grover}} \times \underbrace{N^{1/4}}_{\text{Inner Grover}} \times \underbrace{\log(N)}_{\text{Error Reduction}} \right) = \mathcal{O}(\sqrt{N} \log(N))$$

which up to log-factor achieves the full Grover speedup.

With some cleverness people have since been able to remove the $\log(N)$ factor.

Why couldn't we just do Grover's algorithm once, over the whole table? Well, just because there's a 0 somewhere in the table doesn't mean that there couldn't be a row of all 1's somewhere else.

The first problem in quantum computing that Professor Aaronson worked on was trying to generalize the BBBV Theorem to show that "recursive Grover" is optimal for the OR-of-ANDs. The obvious lower bound is only $\sim N^{1/4}$. After a whole summer trying to solve this problem using the "polynomial method," it remained unsolved. Shortly afterward, Andris Ambainis, then a PhD student at Berkeley, invented a totally new method for proving lower bounds on quantum query complexity, and applied it to solve this problem.

OR of ANDs Tree

We could easily generalize the scheme we used for the OR of ANDs problem to evaluate (e.g.) an OR of ANDs of ORs by doing three recursive layers of Grover search and so forth. If we allow an arbitrary number of layers then we enter a setting commonly seen in A.I. research: *game trees* for two-player

games of alternation such as chess and Go. In this setting the goal is to find a move you can make (represented by an OR over various options), such that given any move that your opponent makes (represented by ANDs over various options), there is a move that you can make in response, etc. . . that eventually wins you the game.

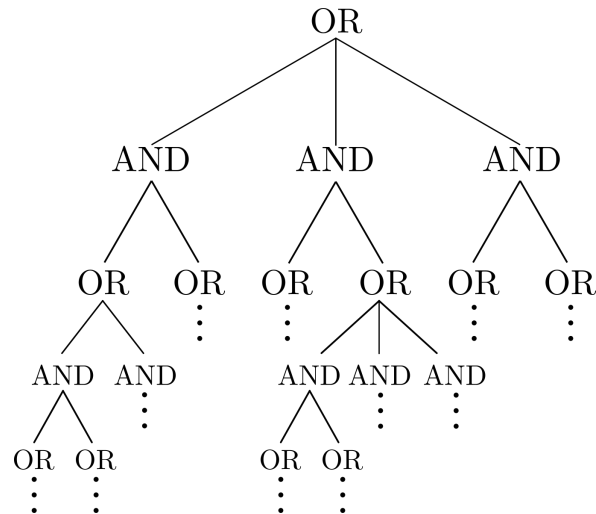


Figure 23.3: An example of an OR of ANDs tree.

The problem is that as the game tree gets deeper and deeper, the advantage of Grover's algorithm over classical search seems to get weaker and weaker. This is for two reasons: first, the amplification that's needed at each layer to prevent error buildup and second, the constant factors, which multiply across the layers. With regards to the second reason you might object that the constant factor for Grover's algorithm is $\frac{\pi}{4} < 1$, so if we multiply this constant factor across layers you'd think we'd do better and better with increasing depth! Note however that each layer actually needs to run Grover's algorithm on the layer below it twice: once to evaluate $|x\rangle \rightarrow |x\rangle |f(x)\rangle$ for that layer, and a second time to uncompute any garbage generated (which would have the effect of destroying the interference effects needed for higher levels of the recursion). So the constant factor actually becomes $\frac{\pi}{2} > 1$.

In short, we still haven't answered the natural question: *can a quantum computer help you play chess?* For game-tree search with a deep enough tree, Prof. Aaronson and others conjectured that the diminishing returns from Grover's algorithm would end up negating any asymptotic advantage over a classical computer.

In 2007, however, Farhi, Goldstone, and Gutmann along with others who

built on their work dramatically refuted that conjecture. The upshot is that we now know how to evaluate *any* game tree with N leaves, no matter how deep, in $\mathcal{O}(\sqrt{N})$ time on a quantum computer. This is also known to be asymptotically optimal.

So, yes, quantum computers probably *would* help you play chess! (At least is brute-force game tree evaluation is a component of the classical algorithm you're competing against). To attach some numbers to this claim, Claude Shannon famously estimated the number of possible board positions in chess as $\sim 10^{43}$, which is certainly out of range for any existing computer on earth. But if quantum computers brought that down to $\sim 10^{21.5}$, solving chess might *just* be doable.

Though it raises a philosophical question: have you actually "solved" chess if you don't have a solution table that anyone can examine, but only a quantum computer that always wins?

Lecture 24: More Applications of Grover’s Algorithm and Quantum Complexity Theory

24.1 More Applications of Grover’s Algorithm

24.1.1 The Collision Problem

In the simplest version of the so-called **Collision Problem**, we’re given quantum black-box access to a function $f : \{1, \dots, N\} \rightarrow \{1, \dots, N\}$, where N is even, and we’re promised that f is two-to-one. The problem is to find x and y such that $x \neq y$ and $f(x) = f(y)$. Since f is two-to-one, there are lots of collisions to be found and the challenge is just to find one of them. In another version of the problem, we’re promised that *either* f is one-to-one or it’s two-to-one, and the problem is to decide which. Clearly if we could solve the search version then we could also solve the decision version, by simply outputting that f is two-to-one if a collision is found or that f is one-to-one if not. Conversely, any lower bound for the decision version implies the same lower bound for the search version (but there’s no obvious reduction in the other direction).

The collision problem often arises in cryptography when we’re trying to break collision resistant hash functions. You can think of the collision problem as being a lot like Simon’s problem but with less structure—or alternatively, as being like the Grover search problem but with more structure.

With a classical randomized algorithm given black-box access to f , $\Theta(\sqrt{N})$ queries are necessary and sufficient to solve the collision problem. *Why?* The upper bound follows from the famous “birthday attack,” which we first saw in

Lecture 18 when discussing Simon's algorithm. If there are N days in the year, then you only need to ask about \sqrt{N} people before there's an excellent chance that you'll find two with the same birthday. This is because what matters is the number of pairs of people, which grows quadratically with the number of people asked. The lower bound can be proven using the union bound. With a random two-to-one function, each pair has only a $\sim \frac{1}{N}$ chance of being a collision, so to find a collision with constant probability you need to look at $\sim N$ pairs or more (and therefore make $\sim \sqrt{N}$ queries).

What about quantumly? Well, we could of course simulate the above randomized algorithm to get $\sim \sqrt{N}$. But there's also a completely different way to get $\sim \sqrt{N}$. Namely, we could first query $f(1)$, and then do a Grover search for an $x \neq 1$ such that $f(x) = f(1)$. So a question naturally arises: *can we combine the two approaches to do even better than \sqrt{N} ?*

In 1997 Brassard, Hoyer and Tapp (BHT) showed how to do exactly that. Here's their algorithm:

- ▶ First, pick $N^{1/3}$ random inputs to f , query them classically and sort the results for fast lookup.
- ▶ Next, run Grover's algorithm on $N^{2/3}$ more random inputs to f (inputs that weren't queried in the first step). In this Grover search, count each input x as marked if and only if $f(x) = f(y)$, where y is one of the $N^{1/3}$ inputs that was already queried in the first step. This requires lookups to our sorted list, but no additional queries to f .

This algorithm makes $N^{1/3} \times N^{2/3} = N$ pairwise comparisons and the runtime is

$$N^{1/3} + \sqrt{N^{2/3}} = \mathcal{O}(N^{1/3}) \quad (24.1)$$

The centerpiece of Prof. Aaronson's PhD thesis was showing that you can't improve on this by much. It was later shown by Yaoyun Shi that you can't improve on it at all.

The BHT algorithm gives a good illustration of one way quantum algorithms can end up with weird running times. You have two or more phases of the algorithm that you try to balance against each other in order to minimize the total time.

At a high level you can see why the BBBV proof that we used to prove the optimality of Grover's algorithm doesn't work for the collision problem. In the BBBV proof we changed a single element from 0 to 1 and then showed that

it would take many iterations for the algorithm to notice. With the collision problem, by contrast, the key issue is that turning a one-to-one function into a two-to-one function requires changing half the elements. Instead, Aaronson and Shi used polynomial approximation theory (a branch of mathematics) to rule out super-fast quantum algorithms for the collision problem.

In some sense, proving a quantum lower bound for the collision problem should be harder than proving one for the Grover problem, because if the lower bound for collision did too much then it would rule out things like Simon's algorithm or Shor's algorithm. The details of the proof are beyond the scope of this course, but generally what allows a lower bound for the collision problem is that it has permutation symmetry that Simon's and Shor's problems lack: if you take a one-to-one function and permute its inputs and outputs arbitrarily, then it's still a one-to-one function, and likewise for a two-to-one function.

24.1.2 Element Distinctness

In addition to the collision problem, there's also the closely related problem of **Element Distinctness**. Here you're given black-box access to the function $f : \{1, \dots, N\} \rightarrow \{1, \dots, N\}$, with no promises about f . The problem is to determine if f is one-to-one. In other words, *are there any duplicates/collisions?*

Classically, the optimal strategy is to hash the elements (or sort them, or use a binary search tree, etc...), which would take N queries plus the amount of computation required for hashing/sorting (say, $N \log(N)$ steps).

Quantumly, there's an algorithm that takes only $\mathcal{O}(N^{3/4})$ queries, as shown in a paper from Buhrman et al. from 2000. Given a list of the N values of the function we split them into \sqrt{N} blocks of \sqrt{N} values each. We then pick a block at random and query all elements in it. If we find a collision in the block, you're already done! If we don't, then sort the elements in the block for fast lookup. Next, do a Grover search on the other items in the list, counting an item as marked if and only if it equals an element from the first block we've queried. As long as we were lucky enough to pick a block that contains at least one element of a collision pair, this algorithm will find such a pair with constant probability. Hence it succeeds with $\frac{1}{\sqrt{N}}$ probability overall. This algorithm uses $\mathcal{O}(\sqrt{N})$ queries and we'll be using it as a subroutine.

In order to achieve a constant success probability (rather than the $\frac{1}{\sqrt{N}}$ probability we found for the algorithm above) naively we'd need to repeat the subroutine above $\mathcal{O}(\sqrt{N})$ times. We can do better than that though. In order to do so we'll introduce an outer layer of Grover search that searches through the \sqrt{N} different starting blocks we could've chosen among, as the initial

block we query in the subroutine above. We'll count a block as "marked" if and only if the inner subroutine finds a collision involving that block. The total runtime for this algorithm is (ignoring for the moment any log-factors from error reduction)

$$\mathcal{O}\left(\underbrace{\sqrt{N}}_{\text{Inner Subroutine}} \times \underbrace{N^{1/4}}_{\text{Outer Grover}}\right) = \mathcal{O}(N^{3/4})$$

Here is high-level summary complete algorithm:

Inner Subroutine:

- ▶ Given a list of the N values of the function we randomly split them into \sqrt{N} blocks of \sqrt{N} values each.
- ▶ We then pick a block at random and query all elements in it.
 - Sort elements for fast lookup. This doesn't require any extra queries.
 - If we find a collision, return it and halt.
- ▶ If we don't find a collision then use Grover to search for collisions between the block we initially selected and the rest of the list. If we find a collision, return it and halt.

Outer Grover:

- ▶ Run Grover over the choice of which of the \sqrt{N} blocks we query initially when running the inner subroutine above.
 - If any of the inner subroutines being Grover searched over return a collision, then return it and halt.

What's the lower bound on Element Distinctness? As a baseline, we know the query complexity has to be at least that of searching a list for a given i , which is $\sim \sqrt{N}$. Pinning down the query complexity of Element Distinctness between \sqrt{N} and $N^{3/4}$ was an open problem for several years. As it turns out, the answer is $N^{2/3}$.

First, Yaoyun Shi noticed that an $\sim N^{2/3}$ lower bound follows from the $\sim N^{1/3}$ lower bound for the collision problem. Suppose for a contradiction that we could solve Element Distinctness in $t = o(N^{2/3})$ queries. The claim is this would let us solve the collision problem in $t = o(N^{1/3})$ queries. *How?* Given a two-to-one function f , pick \sqrt{N} inputs uniformly at random. Since, by the birthday paradox, we can expect to find a collision within that set of

inputs with constant probability, we now simply run our hypothesized Element Distinctness algorithm on that subset. This gives a query complexity of $o((\sqrt{N})^{2/3}) = o(N^{1/3})$, which contradicts the lower bound we proved for the Collision Problem.

Matching this lower bound, in 2003 Andris Ambainis found a quantum algorithm that solves Element Distinctness with $O(N^{2/3})$ queries. His algorithm uses “quantum walks,” which are vaguely like Grover’s algorithm but more sophisticated. It also requires a huge amount of workspace qubits, namely $\sim N^{2/3}$ of them. Whether this large number of workspace qubits is necessary remains open to this day.

24.2 Parity Lower Bound

Given a string $x \in \{0, 1\}^n$, suppose we just want to determine the parity of the string $x_1 \oplus x_2 \oplus \dots \oplus x_n$. That is, whether the total number of 1 bits is odd or even.

Classically, of course, this requires n queries. Quantumly, we’ve seen that we can do it in $\frac{n}{2}$ queries, by splitting x into $\frac{n}{2}$ pairs and then applying the Deutsch-Jozsa algorithm separately to each pair. A beautiful result shows that this is optimal: any quantum algorithm for parity requires $\frac{n}{2}$ queries. This was shown to be the case using a technique known as the polynomial method by Beals et al. in 1998.

As a complement to Parity, it’s also worth briefly discussing the n -bit Majority function, which outputs 0 or 1 depending on whether the input string has more 0’s or 1’s respectively. The quantum query complexity of Majority turns out to be order n —i.e., there is no asymptotic quantum speedup for this problem. This lower bound can also be proved using the polynomial method. However, there is a quantum speedup for a problem closely related to Majority.

For a poll to be accurate within an error of ϵ , how many people do you need to query classically? If you imagine that this poll has two options and that people’s preferences correspond to either 0 or 1, then this is equivalent to approximating the Hamming weight (i.e., number of 1’s) of an n -bit input string to within an additive error $\pm \epsilon n$. Classically you can do this by sampling $\sim \frac{1}{\epsilon^2}$ uniformly random bits and taking their average; this is also tight.

This fact is extremely useful to know when, e.g., choosing the sample size for a political poll to achieve a desired margin of error like, say $\pm 3\%$.

Quantumly it turns out that we can solve this problem using only $\frac{1}{\epsilon}$ queries, a quadratic speedup, by using a clever application of Grover's algorithm (we omit the details here).

24.3 Quantum Complexity Theory

To conclude this lecture let's talk a bit about **Quantum Complexity Theory**, the generalization of computational complexity theory to the quantum realm. That way we can understand the broader context of the quantum algorithms we've seen.

Classically we define complexity classes such as:

- ▶ **P** (Polynomial-Time)– the class of decision problems solvable by a standard, deterministic digital computer in polynomial time.
 - Examples: linear programming, connectivity of graphs, primality testing. . .
- ▶ **NP** (Nondeterministic Polynomial-Time)– the class of decision problems for which there's a deterministic polynomial-time algorithm to verify yes-answers.
 - Examples: factoring, graph isomorphism. . .
- ▶ **NP-hard** problems– roughly speaking, problems to which every **NP** problem can be reduced in polynomial time. In particular, if you had an oracle to solve some **NP-hard** problem in polynomial time, then you could use it to solve everything in **NP** in polynomial time.
- ▶ **NP-complete** problems– those that are both in **NP** and **NP-hard**. Informally, they're “the hardest problems in **NP**”.
 - Examples: Traveling Salesman, 3SAT, Max Clique, Bin Packing, VLSI layout, Sudoku, Super Mario. . .

This picture already involves an enormous mathematical unknown. Famously, no one has ruled out the possibility that $\mathbf{P} = \mathbf{NP}$, in which case all **NP** problems—and in particular, all **NP-complete** problems—would be solvable in polynomial time.

Where does quantum computing fit in? In 1993 Bernstein and Vazirani defined the complexity class **BQP** (Bounded-Error Quantum Polynomial-Time) as a quantum generalization of **P**. Loosely speaking, **BQP** contains all decision problems that can be solved in polynomial time with a quantum computer. *How does **BQP** relate to classical complexity classes?*

We know that $\mathbf{P} \subseteq \mathbf{BQP}$, because Toffoli gates can simulate AND, OR, and NOT gates and hence universal classical digital computation. So any classical digital calculation can be simulated by a quantum computer. We also know from Shor's algorithm that Factoring (when suitably phrased as a decision problem) is in \mathbf{BQP} , though it's not known (to put it mildly) whether Factoring is in \mathbf{P} . We also don't know whether $\mathbf{NP} \subseteq \mathbf{BQP}$: that is, whether quantum computers can solve all \mathbf{NP} problems (including \mathbf{NP} -complete problems) in polynomial time. The BBBV Theorem does tell us that there isn't an "easy" proof of $\mathbf{NP} \subseteq \mathbf{BQP}$ to be had, one that just treats the \mathbf{NP} problem as a black box.

Can quantum computers solve \mathbf{NP} -complete problems in polynomial time? remains one of the big open problems of quantum complexity theory. Of course, if $\mathbf{P} = \mathbf{NP}$ then $\mathbf{NP} \subseteq \mathbf{BQP}$, so any proof of $\mathbf{NP} \not\subseteq \mathbf{BQP}$ would require proving $\mathbf{P} \neq \mathbf{NP}$ at the least. No reduction or equivalence between the $\mathbf{P} = \mathbf{NP}$ and $\mathbf{NP} \subseteq \mathbf{BQP}$ problems is currently known.

We could also ask the converse, *Is $\mathbf{BQP} \subseteq \mathbf{NP}$? In other words, for every problem that a quantum computer can solve, is there a short proof of the answer that's easy to verify classically?* It's possible that there are counterexamples to this, but we don't have any good candidates right now—that is, unless we broaden the definitions of \mathbf{BQP} and \mathbf{NP} beyond decision problems, to capture more general problems such as promise problems, search problems and sampling problems.

The last important question to ask here is: *if \mathbf{BQP} doesn't seem to be contained in \mathbf{P} , and maybe not even in \mathbf{NP} , then what is it contained in? What classical class gives an upper bound on what a quantum computer can do?* Well, Bernstein and Vazirani showed that it's possible to simulate a quantum computer classically with exponential time and polynomial memory, basically by writing an amplitude of interest as a sum of exponentially many contributions, and then evaluating the contributions one by one, reusing the same memory each time and adding the results to a running total. This gives us an upper bound: $\mathbf{BQP} \subseteq \mathbf{PSPACE}$, where \mathbf{PSPACE} (Polynomial Space) is the class of problems solvable on a digital computer using a polynomial amount of memory, but possibly exponential time. It's possible to get a better upper bound on \mathbf{BQP} , but it involves other complexity classes that we won't define here.

So, what would it take to prove that \mathbf{P} is different from \mathbf{BQP} ? Of course this would follow if factoring wasn't in \mathbf{P} , but proving the latter would require showing $\mathbf{P} \neq \mathbf{NP}$! It's also not clear that there is any better hope for proving $\mathbf{P} \neq \mathbf{BQP}$ in the near future than there is for proving $\mathbf{P} \neq \mathbf{NP}$. The reason is that \mathbf{BQP} is sandwiched between \mathbf{P} and \mathbf{PSPACE} :

$$\mathbf{P} \subseteq \mathbf{BQP} \subseteq \mathbf{PSPACE}.$$

For this reason, any proof of $\mathbf{P} \neq \mathbf{BQP}$ would also need to show that $\mathbf{P} \neq \mathbf{PSPACE}$, which is a big unsolved problem in itself.

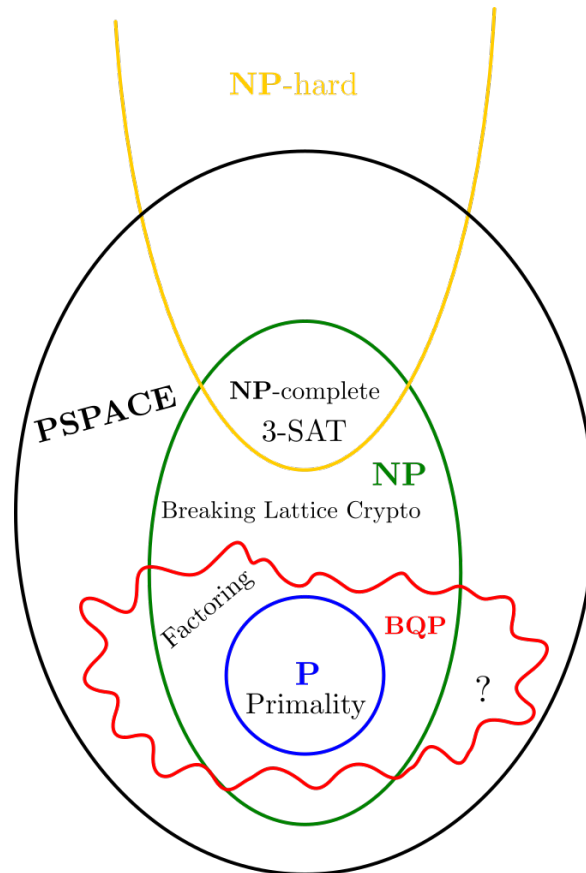


Figure 24.1: Where **BQP** fits in with classical computational complexity classes, as best as we know it today.

Lecture 25: Hamiltonians

25.1 Quantum Algorithms for NP-complete Problems

We've seen how it's an open question whether quantum computers can solve **NP**-complete problems in polynomial time. If this turned out to be possible it would be world-changing.

Like, it would be time for a Manhattan Project to build scalable quantum computers...

But even if it turns out that quantum computers can't solve **NP**-complete problems in polynomial time, the question still remains *how close can they get?* We know from the BBBV Theorem that any approach that ignores the structure of **NP**-complete problems can at most yield the Grover speedup.

*There have been many papers on the preprint repository arXiv.org that claim to solve **NP**-complete problems in polynomial time with a quantum computer, but do so in ways that violate the BBBV Theorem. Virtually all quantum computing papers can be found on arXiv.org, but the site has no peer review so reader beware.*

So to do better than Grover we'd need to exploit problem structure in some way. For example, with Boolean satisfiability we could imagine devising some quantum algorithm that dealt with certain parts of the formula first and worried about other parts later. If we managed to show that any **NP**-complete problem was in **BQP** (i.e., solvable in polynomial time by a quantum computer), then by definition all of **NP** would be in **BQP**. However, if we're talking about small (polynomial) speedups, then the choice of **NP**-complete problem might actually matter, because the process of reduction from one

NP-complete problem to another might cancel out a speed advantage.

The **Adiabatic Algorithm**, proposed by Farhi, Goldstone, Gutmann, and Sipser in 2000 is a famous attempt to achieve a quantum speedup for NP-complete problems (conceivably even an exponential speedup) by actually exploiting their structure. It's an extremely important quantum algorithm, but unlike (say) Shor's or Grover's algorithms, it doesn't come with any rigorous analysis guaranteeing it will run fast in all cases—and indeed, we now know that it doesn't. To this day, no one really knows how useful this algorithm will be in practice.

It's something that people will eagerly experiment with as soon as they have reliable large-scale quantum computers to test it on!

For some instances of optimization problems, the adiabatic algorithm might give a huge speed advantage, but for other instances it gives little or no advantage or is even outperformed by classical algorithms. People are still trying to figure out for which types of instances the algorithm is most useful.

To understand the adiabatic algorithm, we first need to back up and familiarize ourselves with a central concept in quantum mechanics called **Hamiltonians**.

In a physics course on quantum mechanics Hamiltonians would be day-one material and we'd only get to quantum computing and information at the very end (if at all). In this course it's exactly the opposite!

25.2 Hamiltonians

Recall that unitary matrices are discrete-time linear transformations of quantum states:

$$|\psi\rangle \rightarrow U |\psi\rangle$$

But in physics, time is typically treated as continuous. Rather than viewing the transformation of $|\psi\rangle$ to $U |\psi\rangle$ as a discrete jump, it is viewed as the result of a continuous process causing the state to evolve over some interval of time. Hamiltonians are just the instantaneous time generators of unitary transformations. That is, they're things that give rise to unitary transforma-

tions when you “leave them running” for some period of time. Like density matrices, Hamiltonians are described by Hermitian matrices. Unlike density matrices, however, Hamiltonians don’t need to be positive semidefinite or to have trace 1. Physically, Hamiltonians are operators that are used to represent the total energy of a system.

Remember: for H to be Hermitian means that $H = H^\dagger$

From a physics perspective the central equation of quantum mechanics is Schrödinger’s Equation:

$$i \frac{d}{dt} |\psi\rangle = H |\psi\rangle, \quad (25.1)$$

with H being some Hamiltonian. This equation describes the evolution of an isolated quantum pure state in continuous time.

The full version of Schrödinger’s equation also includes the so-called Planck’s constant \hbar , which is needed to convert between units of time and units of energy. But unless we’re dealing with actual experimental data (involving real units like meters, seconds, joules, and so forth) it’s more convenient just to set $\hbar = 1$, the convention we’ll adopt throughout the rest of the course! For future reference, if it ever comes up, the speed of light c is also 1.

Assuming that H is time-independent, we can solve Schrödinger’s equation to find that the state after time t is

$$|\psi(t)\rangle = e^{-iHt} |\psi(0)\rangle. \quad (25.2)$$

Schrödinger’s equation describes a whole system of linear differential equations—one for each coordinate of the vector $|\psi\rangle$ —but we can formally solve it by pretending that the matrix H is a scalar.

25.2.1 Matrix Exponentiation

We need to back up to address a mathematical point: *what does it mean to raise e to the power of a matrix?* The “right” definition turns out to be based

on the standard Taylor series for the exponential function,

$$e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!}, \quad (25.3)$$

where we now just plug in a matrix instead of a scalar in order to get a matrix-valued result. Here are some examples:

$$\exp\left(\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\right) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \exp\left(\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}\right) = \begin{bmatrix} e & 0 \\ 0 & 1 \end{bmatrix} \quad (25.4)$$

More generally, for any diagonal matrix we have

$$\exp\left(\begin{bmatrix} \lambda_0 & & \\ & \ddots & \\ & & \lambda_{n-1} \end{bmatrix}\right) = \begin{bmatrix} e^{\lambda_0} & & \\ & \ddots & \\ & & e^{\lambda_{n-1}} \end{bmatrix}. \quad (25.5)$$

In other words, you can exponentiate a diagonal matrix by exponentiating each diagonal term individually.

The above equation for diagonal matrices might look like a very special case, but for Hermitian matrices it's really all we need! Suppose we're given a Hermitian matrix, A . By the Spectral Theorem, we can write $A = UDU^\dagger$ where D is diagonal and U is some unitary transformation. To compute e^A , we can then write

$$\begin{aligned} e^A &= \sum_{k=0}^{\infty} \frac{(UDU^\dagger)^k}{k!} \\ &= U \left(\sum_{k=0}^{\infty} \frac{D^k}{k!} \right) U^\dagger \\ &= U e^D U^\dagger \end{aligned} \quad (25.6)$$

Thus, in order to exponentiate any Hermitian matrix we simply need to diagonalize it first.

What leverage do we get from H being Hermitian? For what we've said to make sense—and in particular, for it to be consistent with the discrete-time version of QM we've used in the rest of the course—it better be the case that the matrix e^{-iHt} is unitary. Let's now prove this to be true, by using the fact that H is Hermitian.

First claim: If H is Hermitian, then all its eigenvalues are real.

Proof: Suppose λ is an eigenvalue of H . Then by definition, there's some eigenvector $|v\rangle$ such that $H|v\rangle = \lambda|v\rangle$. If we multiply both sides by the corresponding bra $\langle v|$ we get

$$\langle v|H|v\rangle = \lambda$$

If we take the complex conjugate of both sides we get

$$\langle v|H^\dagger|v\rangle = \langle v|H|v\rangle = \lambda^*$$

This implies $\lambda = \lambda^*$, therefore $\lambda \in \mathbb{R}$.

As we stated above, every Hermitian matrix is diagonalizable. This follows directly from the spectral theorem. We won't go through the details of proving the spectral theorem here, but we will make use of an immediate consequence that any Hermitian matrix can be written as

$$H = UDU^\dagger \tag{25.7}$$

where U is a unitary transformation and D is diagonal and real-valued. Now, to show that e^{-iHt} is unitary just diagonalize H :

$$e^{-iHt} = e^{-itUDU^\dagger} = Ue^{-itD}U^\dagger. \tag{25.8}$$

We know from Equation 25.5 that e^{-itD} has the form

$$e^{-itD} = \begin{bmatrix} e^{-it\lambda_0} & & \\ & \ddots & \\ & & e^{-it\lambda_{n-1}} \end{bmatrix}, \tag{25.9}$$

which is clearly unitary because the λ_i 's are reals. Therefore, $e^{-iHt} = Ue^{-iDt}U^\dagger$ is unitary as well.

Note that if $|v\rangle$ is an eigenvector of H associated with the eigenvalue λ then $|v\rangle$ is also an eigenvector of e^{-iHt} with the eigenvalue $e^{-i\lambda t}$. So eigenvectors of H give rise to eigenvectors of the corresponding unitary.

Now, *what about going backwards?* Given a unitary U , can we always find a Hermitian matrix H such that $U = e^{-iHt}$? Yes, this is not hard. First diagonalize U (which we can always do for unitary matrices, again by the spectral theorem) to get $U = VDV^\dagger$. We then just need to take matrix logarithm—like the matrix exponential, this can be defined in terms of the Taylor expansion. Similar to what we found for the matrix exponential, the logarithm of a diagonal matrix, D , can be obtained by taking the logarithm of each entry. For each diagonal element of D , D_{jj} , we find a λ_j such that

$D_{jj} = e^{-i\lambda_j t}$. Will the set of $\{\lambda_j\}$ that we get by solving this be unique? No, because by Euler's formula we can always add $2\pi i$ to the exponent and the equation will still hold. We saw, for example that

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

is a logarithm of the identity matrix. *What else is?*

$$\begin{bmatrix} 2\pi i & 0 \\ 0 & 2\pi i \end{bmatrix}, \begin{bmatrix} 4\pi i & 0 \\ 0 & 6\pi i \end{bmatrix}, \dots \quad (25.10)$$

Thus, any given unitary can arise from infinitely many different Hamiltonians.

25.2.2 Energy

Physicists have a special name for the eigenvalues that you get by diagonalizing a Hamiltonian. They call them *energies*. Note that they're all real and can therefore can be ordered from least to greatest:

$$\begin{bmatrix} \lambda_0 & & \\ & \ddots & \\ & & \lambda_{n-1} \end{bmatrix} \quad (25.11)$$

with $\lambda_0 \leq \lambda_1 \leq \dots \leq \lambda_{n-1}$. To each energy λ_j there corresponds an *energy eigenstate* $|v_j\rangle$ such that $H|v_j\rangle = \lambda_j|v_j\rangle$. *Why are they called energies?* Because they *are* possible values for the system's energy.

Quantum mechanics gives us one explanation for why the concept of "energy" arises in physics. Because unitary matrices arise by exponentiating Hamiltonians and Hamiltonians can be diagonalized and have real eigenvalues.

If we apply the unitary transformation e^{-iHt} to the energy eigenstate $|v_j\rangle$, we get

$$e^{-iHt}|v_j\rangle = e^{-i\lambda_j t}|v_j\rangle, \quad (25.12)$$

meaning that nothing has happened apart from the state picking up a global phase (unobservable by itself) dependent on the energy. The energy eigenstates

form a complete basis (we won't prove that at the moment, but it is indeed true), so we can write an arbitrary state as a superposition over the energy eigenstates:

$$|\psi\rangle = \alpha_0 |v_0\rangle + \cdots + \alpha_{n-1} |v_{n-1}\rangle. \quad (25.13)$$

From this perspective, applying the Hamiltonian H for some amount of t is equivalent to doing

$$e^{-iHt} |\psi\rangle = \alpha_0 e^{-i\lambda_0 t} |v_0\rangle + \cdots + \alpha_{n-1} e^{-i\lambda_{n-1} t} |v_{n-1}\rangle. \quad (25.14)$$

This presents a terrifyingly boring picture of the history of universe! In this picture, all that has ever happened, and all that ever will happen, is that the various energy eigenstates of the universe pick up phases, with each rotating around the unit circle at a speed proportional to its energy.

From the utter lack of interesting activity when we view the world in the energy eigenbasis, we conclude that life is a basis-dependent phenomenon.

But the above picture is actually extremely useful. For one thing, it suggests that we can simply *define* energy as the speed at which a quantum state picks up a phase.

It's not obvious that this corresponds to the usual conceptions of energy in physics, but it turns out that it does. You'll have to go to the physics department for the full details though!

One thing that's clear from our definition is that energy is conserved. More formally, the expectation value of the energy in the state $|\psi\rangle$,

$$\langle\psi|H|\psi\rangle = \sum_j |\alpha_j|^2 \lambda_j,$$

stays the same over time.

We have a special nomenclature for energy eigenstates. The eigenstate $|v_0\rangle$ corresponding to the lowest energy is called the *ground state*, which we'll see plays an extremely special role in the adiabatic algorithm. The corresponding energy λ_0 is the ground state energy. The eigenstate $|v_1\rangle$, corresponding to the second-lowest energy, is called the *first excited state*. The eigenstate $|v_2\rangle$ is called the *second excited state*, and so forth. If there's more than one energy

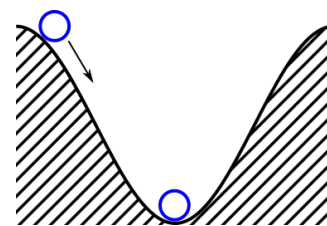
eigenvalue equal to λ_0 , then we get what's called a *ground state degeneracy*. There's no longer a unique ground state, but rather a subspace of two or more dimensions in which every state equally minimizes the energy. We can similarly get degenerate excited states if $\lambda_j = \lambda_{j+1}$ for some larger j . For the most part, though, we'll be able to ignore this.

The standard game plan for much of modern physics goes like this:

- ▶ Write down the Hamiltonian H of your system (or some approximation to H).
- ▶ Diagonalize H .
- ▶ Extract the energy eigenstates.
- ▶ Then, as a first guess, see if your system is just sitting in its ground state doing nothing.

This initial guess that your system is in the ground state turns out to work very well much of the time. *Why are quantum systems often found sitting in their ground states?* Intuitively, it's because physical systems “like” to minimize their energy: when lower energy states are available, they tend towards them. The ground state, by definition, is the lowest they can go. *But since quantum mechanics is time-reversible, how is it even possible for a system to be “attracted” to a certain state?* Excellent question! You can thank the **Second Law of Thermodynamics** and the conservation of energy for this.

The same question arises in classical physics, which is time-reversible too. If you leave a ball rolling around in a basin and return a while later you probably won't find it in an “excited state”—i.e. continuing to roll around. Whatever energy it had in its excited state, it could reach a lower energy by rolling downhill and giving off heat (energy) via friction, eventually coming to a rest.



When this happens, the kinetic energy that used to be in the ball dissipates away in the heat.

In principle it's possible that the reverse could happen: the heat in the basin could coalesce back into the ball and make it spontaneously move. But we essentially never observe that, and the reason comes down to entropy. For all the heat to coalesce back into the motion of the ball would require an absurdly finely-tuned “conspiracy,” in which a massive number of particles synchronize their random motion to impart a kick to the ball. The probability of this synchronized behavior occurring by chance falls off exponentially with

the number of particles. But the reverse process, motion dissipating into heat, requires no similar conspiracy; it only requires that our universe does contain low-entropy objects like balls.

This, in turn, can ultimately be traced back to the low entropy of the universe at the Big Bang—something that no one has satisfactorily explained in terms of anything deeper, but we'll be content to leave it there!

Pretty much exactly the same story works in the quantum case and explains why, when we find quantum systems in nature, they're often sitting in their ground states. If they weren't then their interactions with surrounding systems would tend to carry away excess energy until they were. By contrast, all the quantum algorithms and protocols that we've seen in this course are examples of quantum systems that don't just sit in their ground states. Stuff happens, the system evolves!

We as people don't just sit in ground states either.

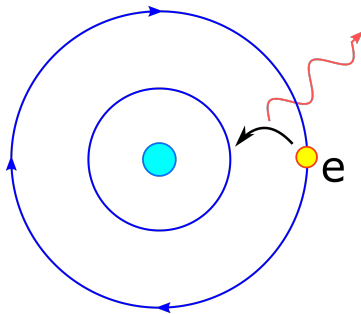


Figure 25.1: Hydrogen atom emitting a photon due and dropping to a lower energy level.

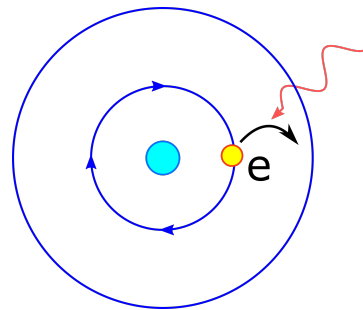


Figure 25.2: Hydrogen atom absorbing a photon and jumping up to a higher energy level.

To give an example of these concepts that the physicists really love, let's talk about the hydrogen atom. In the ground state, a hydrogen atom has its electron sitting in the lowest energy shell (the one closest to the nucleus). The first excited state has the electron in the next shell up (a bit farther away on average from the nucleus). If the atom is in its first excited state, it's easy for it to drop back down to its ground state by emitting a photon. The photon carries away an amount of energy that's exactly equal to the difference between the ground and the first excited energies. Conversely, a hydrogen atom in its

ground state can jump up to its first excited state via the electron absorbing a photon. But the latter process is not spontaneous; it requires a photon just happening to come by. That's why hydrogen atoms in nature are most often found in their ground states.

25.2.3 Tensor Products of Hamiltonians

Just like with unitaries, we can take the tensor product of two or more Hamiltonians to produce a Hamiltonian acting on a larger system:

$$H = H_0 \otimes H_1.$$

We do *not*, in general, have $e^{-i(H_0 \otimes H_1)} = e^{-iH_0} \otimes e^{-iH_1}$. However, that equation *does* hold (up to a global phase) if either H_0 or H_1 is the identity matrix. In that case, we can think of H as acting nontrivially on only one of the two subsystems and trivially on the other (just like with tensor products of unitaries).

25.2.4 Addition of Hamiltonians

Our final topic for this lecture is an important operation that we can do with Hamiltonians, but which we couldn't generally do with unitaries. Namely, we can add them! We can do this simply because the sum of two Hermitian matrices is itself Hermitian, and every Hermitian matrix is a valid Hamiltonian:

$$H = H_0 + H_1 = H_0^\dagger + H_1^\dagger = H^\dagger. \quad (25.15)$$

What does this mean physically? Intuitively, it just means we've got two things going on at the same time. For example, H_0 and H_1 could correspond to two different forces acting on our system.

Building a complicated Hamiltonian by summing local terms could also be seen as somewhat analogous to building a complicated quantum circuit by composing 1 and 2-qubit gates.

To illustrate, at an extremely high level we could write the Hamiltonian for the Standard Model of elementary particle physics as

$$H_{\text{SM}} = H_{\text{Kinetic}} + H_{\text{EM}} + H_{\text{Strong}} + H_{\text{Weak}}$$

where H_{Kinetic} is the Hamiltonian corresponding to the kinetic energy and where H_{EM} , H_{Strong} , and H_{Weak} are the Hamiltonians corresponding to electromagnetism and to the strong and weak nuclear forces respectively. This isn't exactly right for all sorts of reasons, but is good enough to get across the point.

The Standard Model excludes gravity.

If A and B are matrices is it generally the case that $e^{A+B} = e^A e^B$? Alas, no. Indeed it's not hard to find a 2×2 counterexample (we'll leave that as an exercise). On the other hand you can check using the Taylor series definition that if A and B commute (that is, $AB = BA$), then $e^{A+B} = e^A e^B$ does hold.

We'll care a lot about unitary transformations of the form $e^{-it(A+B)}$, or their counterparts with many more than two Hamiltonians being summed. In particular, we'll need a way to apply these sorts of unitaries efficiently given only the ability to apply A and B by themselves—even if A and B don't happen to commute. Fortunately there's a trick for this, known as **Trotterization**. The trick is to use the following approximation:

$$e^{A+B} \approx \underbrace{e^{\epsilon A} e^{\epsilon B} e^{\epsilon A} e^{\epsilon B} \dots e^{\epsilon A} e^{\epsilon B}}_{\frac{1}{\epsilon} \text{ Times}} \quad (25.16)$$

This basically means that we can achieve the same effect as A and B occurring simultaneously by repeatedly switching between doing a tiny bit of A and a tiny bit of B . We won't do it here, but it's possible to prove that the approximation improves as ϵ decreases, becoming an exact equality in the limit $\epsilon \rightarrow 0$. This is important for the question of how to simulate a real-world quantum system using a quantum computer. Indeed, the straightforward approach is just:

- ▶ Discretize all the degrees of freedom (positions, momenta, etc...) that aren't already discrete.
- ▶ Write the total Hamiltonian H acting on the system as a sum of "simple" terms (say, terms that act on only 1 or 2 particles at a time).
- ▶ Trotterize H in order to approximate it by a product of "simple" unitary transformations.

To flesh this out a bit more we ought to say something about what the Hamiltonians of real physical systems tend to look like. Suppose we model the universe as a gigantic lattice of qubits, say in 2 or 3 dimensions (hey, it *is* a quantum computing class!). This is actually exactly the sort of model used to study many condensed matter systems. Suppose too that we only consider

interactions between neighboring qubits. In that case, the total Hamiltonian that acts on the qubits can be written

$$H = \sum_j H_j + \sum_{j \sim k} H_{j,k}. \quad (25.17)$$

Here H_j is a Hamiltonian that acts only on the qubit j and trivially (i.e., as the identity, which has no effect) on all the others, for example:

$$H_0 = \begin{bmatrix} h_1 & h_2 \\ h_3 & h_4 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Meanwhile, for every edge (j, k) in the lattice, H_{jk} is a Hamiltonian that acts nontrivially on the neighboring qubits j and k and trivially on all the other qubits, for example:

$$\begin{bmatrix} 0 & & & \\ & 0 & & \\ & & 0 & \\ & & & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}^{\otimes n-2}.$$

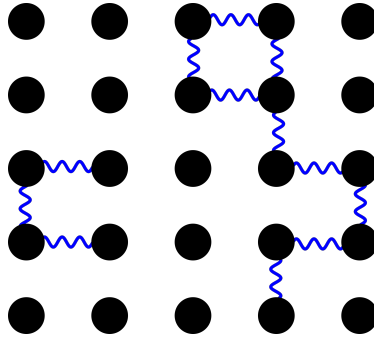


Figure 25.3: Example of a lattice of qubits with some set of pairwise nearest-neighbor interactions (blue squiggly lines) between them.

So each qubit “talks” only to its immediate neighbors in the lattice. Even so, evolving the Hamiltonian over time gives us effects that can propagate arbitrarily far.

As soon as we’ve written this, though, we face a puzzle. *Won’t this lead to faster-than-light travel?* Indeed, even when t is arbitrarily small, one can check that the unitary matrix e^{-iHt} will generally contain effects coupling every qubit in the lattice to every other one. Granted, the magnitude of these effects will

fall off exponentially with distance, but causality demands that there should be literally *zero* effects propagating across the lattice faster than light.

So what's the resolution? Basically it's just that the picture we're using comes from non-relativistic quantum mechanics, so it yields a good approximation only if the relevant speeds are small compared to the speed of light. When the speeds are larger, we need the framework of quantum field theory, which *does* ensure that faster-than-light influences are exactly zero.

OK, now we're ready to set things up for the next lecture. Suppose that H , a Hamiltonian acting on n qubits, is the sum of many "simple" Hamiltonians acting on a few qubits each:

$$H = H_0 + \cdots + H_{m-1}.$$

Since H is a $2^n \times 2^n$, matrix figuring out its ground state (or ground states) by brute-force diagonalization could be extremely time-consuming. This leads to a question: *if I know the ground states of the H_j 's individually, can I combine them in some simple way to get the ground state of H itself?* Alas, the answer is almost certainly "no." More precisely, we claim that finding the ground state of a Hamiltonian of this form is an **NP**-hard problem. To prove this we'll show how to take any instance of the famous 3SAT problem and encode it into the ground state problem. Suppose we have a Boolean formula in n variables,

$$\varphi(x_0, \dots, x_{n-1}) = c_0 \wedge \cdots \wedge c_{m-1}, \quad (25.18)$$

an AND of m clauses, where each clause c_i acts on at most 3 variables. We can now define an n -qubit Hamiltonian H as follows:

$$H = \sum_{i=0}^{m-1} H_i \quad (25.19)$$

The term H_i acts on at most 3 qubits, corresponding to the bits acted on by c_i , and as the identity on all the other qubits. It imposes an "energy penalty" if and only if the qubits are set in such a way that c_i is violated. For example,

Lecture 26: The Adiabatic Algorithm

26.1 Local Hamiltonians

At the end of the last lecture, we saw that the following problem is **NP**-hard: Given as input an n -qubit Hamiltonian H of the form $H = H_1 + \cdots + H_m$, with each H_i acting on at most 3 of the n qubits, estimate H 's ground state energy. We call this problem the **Local Hamiltonians Problem**.

This is a quantum generalization of the 3SAT problem which arises very naturally in condensed matter physics.

How do you give someone a Hamiltonian like that? Providing the full $2^n \times 2^n$ Hermitian matrix would be wasteful. Instead you can simply list the local terms $\{H_0, \dots, H_{m-1}\}$ (to some suitable precision), together with the qubits to which they're applied.

Is this problem **NP**-complete? Since we know it's **NP**-hard, what we're asking here is whether it's also in **NP**. In other words, when we claim that the ground-state energy of some Hamiltonian is at most (say) 5, *can we prove it by giving a short witness?* It turns out that we can—but, as far as anyone knows today, only by giving a quantum witness! A quantum witness that works is simply the n -qubit ground state itself. Thus, the Local Hamiltonians problem is not known to be in **NP**; it's only known to be in the quantum analogue of **NP**, which is called **QMA** (Quantum Merlin-Arthur). An important theorem from around 1999 says that Local Hamiltonians is actually complete for **QMA**, just like 3SAT is complete for **NP**.

*For the specific Hamiltonian H we constructed in the last lecture—the one that encoded 3SAT—there **would** be a short classical witness. Because H was a diagonal ma-*

trix, its ground state is always just a classical basis state. But what if H is non-diagonal and its ground state is some complicated entangled state?

*So, if natural quantum systems like to settle into their ground states, and if finding the ground state is **NP**-hard, does this mean that we could use quantum systems to solve **NP**-hard problems?* People talked about such questions even before the concept of quantum computing was in place. But there's a serious problem. It's not always true that natural quantum systems quickly settle into their ground states. Starting from hard instances of 3SAT might produce complicated and exotic Hamiltonians, far from physicists' usual experience. Those complicated Hamiltonians might be ones for which it's hard to reach the ground state.

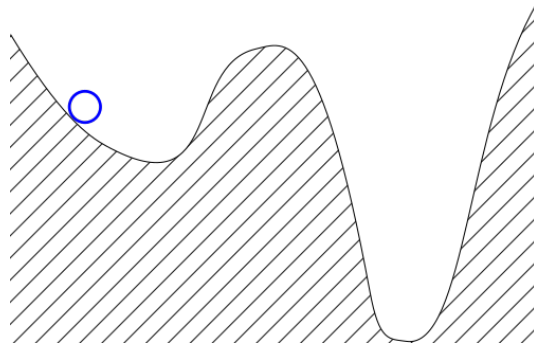


Figure 26.1: Optimization landscape with a local minimum that the particle might get trapped in.

In the hillside above, will the ball get to the point that minimizes its gravitational potential energy? Probably not anytime soon! If we wait a million years, maybe a thunderstorm will push the ball up over the hill in the middle, or something. But for the foreseeable future, it's much more likely for the ball to rest at the local minimum on the left.

In hard real-world optimization problems you may have a very bumpy landscape, with thousands of dimensions and plenty of local optima to get trapped in. You might wonder if quantum computing could help us wade through these local optima, and it certainly seems plausible. In fact, the hope it could was a central starting point for today's topic.

26.2 The Adiabatic Algorithm

This is the last quantum algorithm we'll cover in this course. At the core of the **Adiabatic Algorithm** is a theorem from the 1920s called the **Adiabatic Theorem**. This theorem says the following: suppose you apply some initial Hamiltonian H_i to a system in H_i 's ground state. So far nothing happens, but then suppose you slowly and continuously change H_i into some final Hamiltonian H_f . Provided that the transition was slow enough, the final state of the system you'll end up in is (extremely close to) the ground state of H_f .

Assuming a few fine-print conditions that we won't go into.

So, gradually changing the Hamiltonian moves us from one ground state to another ground state. In the minds of Farhi, Goldstone, Gutmann and Sipser this suggested a plan to solve **NP**-hard problems using Hamiltonians.

Here's how the adiabatic algorithm would work with 3SAT as an example. First we need to pick a Hamiltonian with a known and easy to prepare ground state. For example, the Hamiltonian

$$H = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad (26.1)$$

with eigenstates $|+\rangle$ and $|-\rangle$. The energy of $|+\rangle$ is 0 and the energy of $|-\rangle$ is 2, so $|+\rangle$ is the ground state. We can create an initial Hamiltonian H_i (note we're using i here to denote "initial," not as an index) by applying H to each qubit individually:

$$H_i = \sum_{j=0}^{n-1} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}_j. \quad (26.2)$$

Here the subscript j is meant to denote that the j^{th} term acts nontrivially on qubit j , tensored with the identity matrix on the remaining $n - 1$ qubits. The ground state of H_i , namely $|+\rangle^{\otimes n}$, has the advantage that it is easy to prepare on a quantum computer. We then gradually change H_i to another Hamiltonian H_f , which encodes some n -bit 3SAT instance that we'd like to solve:

$$H_f = \sum_{j=0}^{m-1} h_j, \quad (26.3)$$

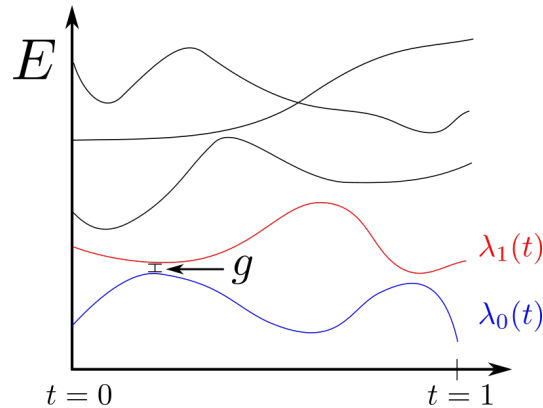


Figure 26.2: Potential plot of energy eigenvalues as a function of time in the adiabatic algorithm.

where each of the h_j 's in H_f encodes one of the m clauses, imposing an energy penalty on non-satisfying assignments. In the simplest version of the algorithm, we interpolate between H_i and H_f linearly, by adjusting a time parameter $t \in [0, 1]$ as follows

$$H_t = (1-t)H_i + tH_f. \quad (26.4)$$

Since each H_t is a sum of terms acting on a few qubits only we can apply these H_t 's using Trotterization (discussed in the last lecture). If everything goes according to plan we'll end up in the ground state of H_f —in other words, the optimal solution to our 3SAT instance. *So what's the catch?* Well, the crux is that the transition from H_i to H_f must be sufficiently slow, but we haven't defined what we mean by "sufficiently slow" yet. *How slow do we need to go for this to work?*

To build intuition, let's plot an example of what the eigenvalues of H_t might look like as a function of the time parameter t (see Figure 26.2). H_t could have as many as 2^n eigenvalues. We're especially interested in the smallest eigenvalue (the ground energy) and the one right above that (the first excited energy). Our goal is to stay in the ground state throughout the entire evolution. The eigenvalues, also called energy levels, can change continuously over time. Sometimes two energy levels can even cross each other. Physicists call that a *level crossing*. When they almost cross it's an *avoided level crossing*. If the two lowest energies cross each other, then we leave the ground state. Even if the two lowest energies barely avoid crossing each other there's a significant risk that we'll leave the ground state. The closer together the two lowest energies get, the slower we have to run the algorithm to avoid leaving

the ground state. We define the **Minimum Eigenvalue Gap**, g , as the minimum of the energy difference between the first excited energy and the ground energy as a function of the time t .

$$g = \min_{t \in [0,1]} (\lambda_1(t) - \lambda_0(t)). \quad (26.5)$$

We'll call the time at which g is minimized t_{\min} . The gap g turns out to be a crucial quantity for determining how long the adiabatic algorithm will take to solve a given problem instance. Roughly speaking, in order to remain in the ground state throughout the entire computation, we need to ensure that near t_{\min} the rate of evolution of the system is $\propto g^2$. This requirement means that the overall computation will run in something like $\sim \frac{1}{g^2}$ time.

If we could show that g was always lower-bounded by $\frac{1}{n^{O(1)}}$ for 3SAT (or some other **NP**-complete problem), then we'd get **NP** \subseteq **BQP**; quantum computers would be able to solve all **NP** problems in polynomial time. In reality we're approximating the Hamiltonians with discrete-time unitary transformations so we'd end up with something *close* to the ground state of H_f , not the ground state itself. But even that would still be good enough to solve our **NP**-hard problem.

So the question boils down to: *what is the behavior of the minimum spectral gap for the problem instances that we want to solve?*

Ed Farhi once went to an expert in condensed matter physics and asked "this is the system we're looking at, do you think that g will decrease polynomially or exponentially as a function of n ?" The expert said: "I think it will decrease exponentially." That's not the answer Farhi wanted to hear. So Farhi asked "why? What's the physical reason?" After thinking about it some more, the expert responded "because otherwise your algorithm would work."

What emerged after a couple of years is that, for hard instances of 3SAT (or even 2SAT, for that matter), the minimum eigenvalue gap often *does* get exponentially small. At the avoided level crossing you'd need to run the algorithm for an exponential number of steps in order to remain in the ground state and thereby solve your SAT instance.

But the story doesn't end here. The physicists regrouped and said, "Okay, so maybe this technique doesn't always work, but it might still give big advantages for *some* types of optimization problems!"

By now there's been lots of research on classifying the types of solution

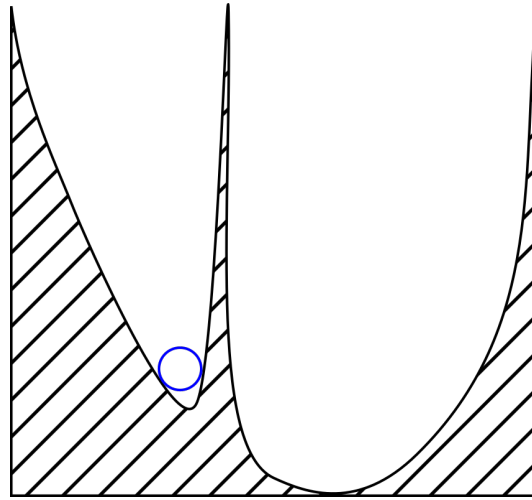


Figure 26.3: Optimization landscape where a sharp thin peak separates the global minimum from the rest of the landscape.

landscapes where the adiabatic algorithm performs well and those where it performs poorly. Some encouraging results came from Farhi, Goldstone, Gutmann in 2002. These authors constructed landscapes that had a global minimum at the bottom of a wide basin, but also a tall thin spike blocking the way to that minimum. Starting from the far left a classical algorithm based on steepest descent would get stuck forever at the base of the spike (i.e., at a local minimum) and would never reach the global minimum.

OK, but before we examine the performance of the adiabatic algorithm on this sort of landscape *shouldn't we first look at better classical algorithms?* Indeed, one example of such an algorithm is **Simulated Annealing** which, much like the adiabatic algorithm, will always eventually reach the global minimum if you run it for long enough. In some regards simulated annealing can be thought of as a classical counterpart to the adiabatic algorithm.

The basic idea of simulated annealing is to evaluate the fitness function around the current point and then:

- ▶ Most of the time: Make a move that improves fitness.
- ▶ Some of the time: make a move that worsens fitness.

The probability of making a move that worsens the fitness is time-dependent and decreases as time goes on. Simulated annealing makes a trade-off between exploration of the general landscape and focusing in on the current steepest path downwards.

The name “simulated annealing” comes from a technology 7000 years old. Annealing is the process of making a metal stronger by heating it up and then slowly cooling it. This gives the atoms in the metal a chance to bounce around and escape from a local optima that might be causing brittleness and then slowly settle into a better optimum.

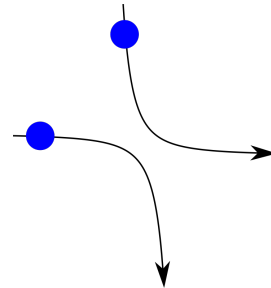
On the fitness landscape with the spike in Figure 26.3, simulated annealing would eventually get over the spike despite how energetically unfavorable it is. However, if the spike is tall enough then it would take an exponential amount of time.

We’d be waiting for the thunderstorm, so to speak.

On the other hand, if the spike is thin enough then, Farhi et al. showed that the adiabatic algorithm can get past it in only polynomial time. It does so by exploiting a well-known quantum phenomenon called **Tunneling**.

Popular articles explain tunneling by saying that a quantum particle can get through barriers that a classical particle could never get through. It would probably be more accurate to say: “in places that a classical particle would need exponential time to get through, sometimes a quantum particle can get through in polynomial time.” In terms of interference, we can say, “The paths that involve the particle not going over the spike interfere destructively and cancel each other out, leaving only paths where the particle does get over it.”

The phenomenon of tunneling is important in many places in physics. For one thing, it’s why the sun can shine! Nuclear fusion requires hydrogen atoms to get super close before they realize that it’s energetically favorable for them to fuse. The trouble is, when they’re not quite so close, they strongly repel each other (because both nuclei are positively charged). When quantum mechanics came along, it explained how, while the energy barrier would prevent fusion classically, it still happens because the nuclei are able to tunnel past the barrier.



Anyway, the 2002 paper of Farhi et al. was good news for the adiabatic algorithm, but tunneling only helps if the spike is sufficiently thin. Since then, we’ve learned more about the types of fitness landscapes for which the adiabatic algorithm is expected to help. In a landscape like the one pictured in Figure 26.4 simulated annealing and the adiabatic algorithm would both have trouble, and would both take an exponential amount of time.

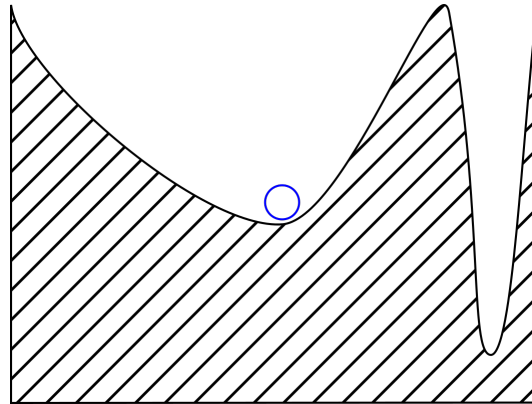


Figure 26.4: An optimization landscape where a very tall thick hill separates the global minimum from the rest of the landscape. The thicker the barrier, the less helpful tunneling is in practice.

Or consider the fitness landscape in Figure 26.5 (we can only draw a 1-dimensional cross-section, but imagine that all of the 2^n solutions in an n -dimensional hypercube have equal values except for the one good solution). This would also take exponential time for both simulated annealing and the adiabatic algorithm to traverse. We actually already know this by the BBBV Theorem—because in this case we’re effectively just querying a black box in an attempt to find a unique marked item.



Figure 26.5: Black-box search style optimization landscape in which only a small region of the landscape is a lower energy than the remaining (flat) landscape.

It turns out that if you’re clever about how you run the adiabatic algorithm, you can achieve the Grover speedup in the case above, but not anything faster. Indeed, the BBBV Theorem tells us that $\Omega(2^{n/2})$ steps are needed using the adiabatic algorithm or any other quantum algorithm. What’s cool is that just by knowing BBBV, without any physics, that the spectral gap has to decrease exponentially.

Just like the expert who Farhi consulted was alluding to with his wisecrack, physicists can use knowledge from quantum algorithms to learn new things about spectral gaps.

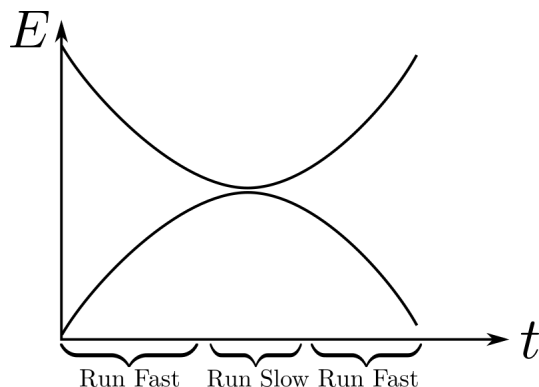


Figure 26.6: High-level sketch of the strategy for getting a Grover speedup using the adiabatic algorithm. The key idea is that we only need to run the algorithm slowly in the vicinity of the minimum eigenvalue gap.

OK, here's a subtler question. Suppose the adiabatic algorithm had worked to solve 3SAT in polynomial time. *Would that have violated the BBBV Theorem?* It turns out that the answer is no. The BBBV Theorem applies only to black-box search. The Hamiltonian encoding a 3SAT instance contains richer information than the black box considered by BBBV. In particular, it encodes not merely whether each possible solution is satisfying or unsatisfying, but also the *number* of clauses that it violates. A 2002 paper by van Dam, Mosca, and Vazirani showed that that information alone is enough to reconstruct the 3SAT instance in polynomial time—and hence also to solve the instance in polynomial time if we assumed (for example) that $\mathbf{P} = \mathbf{NP}$! This means that there's no hope of proving a black-box lower bound like BBBV in this setting.

We also know classical algorithms that can solve 3SAT in less than $2^{n/2}$ time, the best currently known is $\mathcal{O}(1.3^n)$. This is another way of seeing that the BBBV Theorem can't encompass everything that it's possible to do on 3SAT.

OK, let's consider one more type of landscape: the type pictured in Figure 26.7. Here a funny thing happens: simulated annealing gets into the local minimum, but the algorithm then escapes it, crosses the plateau and reaches the global minimum in polynomial time. Meanwhile the adiabatic algorithm just keeps returning to the local minimum and takes exponential time to reach the global minimum!

Sometimes a classical algorithm performs better.

There's even further problem with establishing quantum speedups for adi-

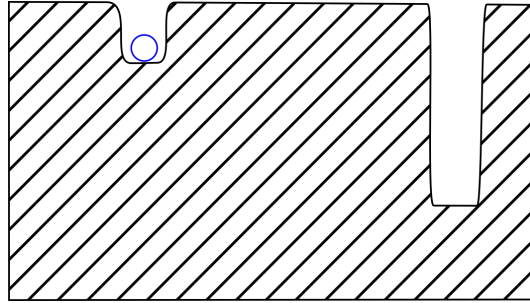


Figure 26.7: The adiabatic algorithm struggles with this sort of optimization landscape while classical algorithms like simulated annealing have little difficulty.

adiabatic optimization, which is that classical computing is a moving target. A classical computer is not limited to simulated annealing or any other specific algorithm. Thus, even if we established that the adiabatic algorithm was exponentially faster than simulated annealing for some class of “real-world” optimization landscapes, we’d still need to compare against other classical algorithms, which might exploit detailed knowledge about the landscapes in question. Particularly relevant here is the Quantum Monte Carlo (QMC) algorithm—which, despite its name, is a quantum-inspired algorithm that runs on a *classical* computer. QMC is widely used in many-body physics. We won’t explain QMC in any detail but will simply say that, even in the artificial cases where the adiabatic algorithm exponentially outperforms simulated annealing, recent work suggests that QMC can often match the adiabatic algorithm’s asymptotic performance classically (albeit often with a much larger constant prefactor). Research continues; there are other cases where adiabatic optimization appears to do better than QMC.

So, what about the optimization landscapes that arise in real-world, industrial applications? Do there or don’t there exist any that the adiabatic algorithm can traverse exponentially faster than any classical algorithm? The truth is, we really don’t know yet. Reaching a consensus on this might require building a scalable quantum computer and then testing the algorithm out!

Which brings us to our final point about the adiabatic algorithm. We’ve talked about implementing the adiabatic algorithm on a conventional, gate-based quantum computer, by using Trotterization to approximate Hamiltonians by discrete sequences of gates. But you might complain that that approach seems roundabout. Since the underlying physics that describes our qubits is based on Hamiltonians anyway, *why not just directly map the adiabatic algorithm onto the continuous-time evolution of the qubits and skip the Trotter-*

ization part? Indeed, the adiabatic algorithm could be seen not only as an algorithm but also as a proposal for the physical implementation of quantum computers. An important result by Dorit Aharonov et al., from 2004, says that adiabatic quantum computers would be universal for quantum computation; that is, able to solve all **BQP** problems efficiently.

There's a venture-capital-backed startup called D-Wave that's been building special-purpose devices to implement a noisy approximation to the adiabatic algorithm (called quantum annealing), using physical Hamiltonians themselves. D-Wave's latest model has about 2000 superconducting qubits. You can encode an optimization problem of your choice onto their chip by choosing the interaction Hamiltonian for each pair of neighboring qubits.

D-Wave was all over the press because they actually sold a few of their machines to companies like Google and Lockheed Martin and were notorious for claiming that quantum computing is “already useful in practice.” They were even on the cover of Time magazine!

Professor Aaronson has been to D-Wave's headquarters. Funnily enough their machine is literally a room-sized black box (most of the hardware inside the box is devoted to cooling; the actual qubits are on a chip no larger than an ordinary computer chip).



Figure 26.8: D-wave's room-sized 2000-qubit quantum annealing device.

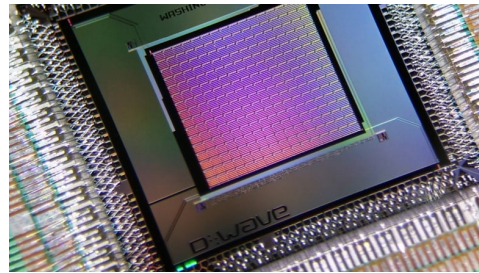


Figure 26.9: Close-up view of one D-Wave's quantum annealing chips.

So what's the verdict? Experimental data shows that the D-Wave device is indeed able to solve optimization problems encoded in its special format at a speed that's often competitive with the best known classical algorithms.

Unfortunately, results over the past decade do not clearly show any quantum speedup over the best classical algorithms. *Why not?* Roughly speaking there are three main possible causes for the lack of speedup on D-Wave’s current devices. As far as we know, the truth might be any combination of them.

- ▶ Inherent limitations of the adiabatic algorithm.
 - Even if we had a perfect quantum computer, running at absolute zero temperature, the minimum spectral gaps might simply be exponential small for almost all interesting optimization problems—in which case, the adiabatic algorithm could provide only limited speedups even in theory.
- ▶ Limitations in the quality of D-Wave’s qubits.
 - Even if the minimum spectral gap were inverse-polynomial, it still probably wouldn’t be constant. That is, it would presumably shrink to zero as a function of the input size. This is a problem for the following reason: it turns out that, if the temperature of the qubits is such that the average amount of thermal energy is of the same order as the minimum eigenvalue gap, then we’ll typically see level crossings *even if we run the adiabatic algorithm at the “right” speed*. The D-Wave qubits are cooled to about 10 milliKelvin. By normal standards that sounds extremely cold, but it might not be cold enough to avoid level crossings on instances of interesting sizes! What one really wants here is absolute zero—but of course, the cooling cost would increase enormously as one approached that, eventually becoming prohibitive.
- ▶ Stoquastic Hamiltonians.
 - A Hamiltonian H is called *stoquastic* if all its off-diagonal terms are real and non-positive. One can show that if H is stoquastic, then H has a ground state involving nonnegative real amplitudes only. As a result, it turns out that stoquastic Hamiltonians are often much easier than arbitrary Hamiltonians to simulate on a classical computer. For example, the QMC algorithm mentioned earlier tends to work extremely well for approximating the ground states of stoquastic Hamiltonians, and less well for non-stoquastic Hamiltonians. Unfortunately D-Wave’s hardware has been limited to applying stoquastic Hamiltonians only. Thus, even if quantum annealing were able to yield an exponential speedup at a fixed nonzero temperature (like 10 milliKelvin), it might be that it could only do so with non-stoquastic Hamiltonians.

This sounds pretty bad! *Why is anyone optimistic that quantum computing could scale even in principle?* We'll see why in the next lecture when we explore the basics of quantum error-correction, a technique that has not yet been demonstrated (by D-Wave or by anyone else), but that many researchers expect will ultimately be necessary for scalable quantum computing.

Lecture 27: Quantum Error Correction

At the end of the last lecture we discussed some of the difficulties with achieving a quantum speedup using currently available quantum computing devices such as the one manufactured by D-Wave's. We saw how D-Wave's devices are cooled to 10 milliKelvin, but even that might be too hot and lead to too much decoherence and error! In the setting of adiabatic quantum computing this shows up mostly as unwanted level crossings. Based on this you might wonder *if even 10 milliKelvin isn't cold enough—if nothing short of absolute zero and perfect isolation seem to suffice—then why should building a scalable quantum computer be possible at all?*

We need to separate two issues. First, there's the “engineering challenge” of building a scalable quantum computer. Everyone agrees that at a bare minimum it will be staggeringly hard to achieve the required degree of isolation for thousands or millions of qubits when those qubits also need to interact with each other in a carefully choreographed way. Maybe various practical problems will prevent human beings from doing it in the next 50 or 100 years. Maybe it will be too expensive. Theory alone can't answer such questions. Then there's the question of whether anything prevents scalable quantum computing even in principle. If quantum mechanics itself were to break down that could certainly prevent quantum computing from being possible—but it would also represent a much more revolutionary development for physics than “merely” building a quantum computer! *So, short of a breakdown of quantum mechanics, on what grounds have people argued that scalable quantum computers aren't possible?*

- ▶ Large entangled states are inherently fragile, so they might be impossible to maintain.
 - If only one qubit in the computer decoheres then the entire system could decohere.
 - If only one qubit in a “Schrödinger cat” type state leaks out into

the environment the quantum coherence between the $|\text{Alive}\rangle$ and $|\text{Dead}\rangle$ components is destroyed.

- ▶ Applying unitary gates may produce lots of errors, which will snowball over time.
 - Maybe your $R_{\pi/4}$ gate rotates by 46° instead of 45° . If so, then over many applications of the gate the errors would pile up.

If you pick up a CS textbook from the 1950's you'll see plenty of discussion surrounding "analog vs digital computers". The disappearance of analog computers from the scene had much to do with the difficulty of correcting continuous errors. Is a quantum computer simply another kind of analog computer? We'll see the answer shortly.

- ▶ The No-Cloning Theorem presents inherent limits to error-correction.
 - Classically the most simple way to deal with errors is by repetition. Because of the No-Cloning Theorem it seems that we can't do the same with quantum computers.

However, despite these potential problems there were fundamental discoveries in the mid-1990s that addressed all three of these concerns. These discoveries convinced most physicists and computer scientists that, short of some revolutionary change to known physics, the difficulties of building a scalable QC are "merely" difficulties of engineering and not of principle.

27.1 Classical Error Correction

Before discussing **Quantum Error Correction** let's briefly review how error correction works classically.

You could take a whole course about this subject. We'll do the 5-minute version here.

The most simple form of classical error correction is based on the idea of introducing redundancy by directly using repetition. The simplest repetition-based error correcting code is the **3-bit Repetition Code** which lets us encode one logical bit using 3 physical bits. We encode a logical 0, denoted $\bar{0}$, as 000. Likewise we encode a logical 1, denoted $\bar{1}$, as 111. We call the bit

strings selected to encode the logical states the *codewords*. We claim that this code lets us both detect and correct an error in any one physical bit. Given an 3-bit input string $x = x_0x_1x_2$ we do *error detection* by checking whether $x_0 = x_1 = x_2$. Assuming that at most one bit experiences an error we'll be able to identify it. If we detect an error (say in x_0) we can do error correction by setting $x = \text{MAJORITY}(x_0, x_1, x_2)$.

It can be shown that any code that can both detect and correct a single bit-flip error must use at least 3 bits. By contrast, if we just want to detect a single bit-flip error and not correct it 2 bits suffice.

More sophisticated codes are able to encode many logical bits at once, not just a single bit, while detecting and correcting a large number of errors (say, any 10% of the physical bits being flipped). In this lecture, though, we'll focus on encoding just a single bit (or qubit) and protecting against just a single error in order to get the main conceptual points across.

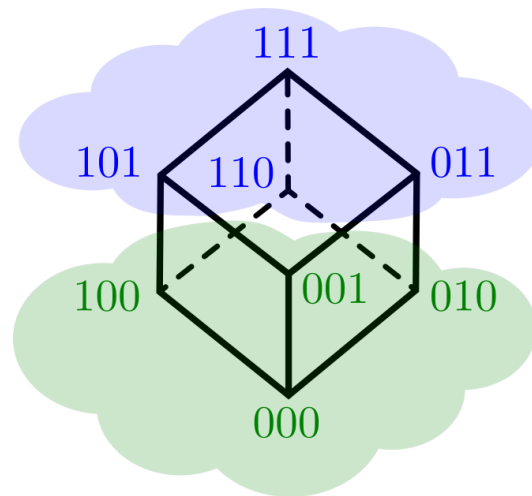


Figure 27.1: Graphical representation of the 3-bit bit-flip code. Each corner of the cube is labeled with a particular 3-bit string, with adjacent corners differing in exactly one bit. The bottom (green) cloud encloses the set of states reachable from 000 when at most one bit-flip occurs. The upper (blue) cloud encloses the set of states reachable from 111 when at most one bit-flip occurs. In order for our error correcting code to work, the set of states in the two clouds cannot have any overlap.

A useful geometric picture for why the 3-bit repetition code works is given in Figure 27.1. We can associate each of the 8 3-bit strings with a corner of a cube selected such that each point differs from its neighbor in exactly one place (the hamming distance is 1). Essentially, the code simply picks two points on the cube that are maximally far from each other and declares one to be the encoding of 0 and the other to be the encoding of 1. This idea generalizes to codes with longer codewords, in which case the set of possible bit strings correspond to points on the boolean hypercube.

000 and 111 can each get corrupted to any point in their respective “clouds”, but since the two “clouds” don’t overlap we’re able to correct the error. We’ll seek to replicate this behavior in the quantum case.

27.1.1 Classical Fault-Tolerance

In the early days of classical computing there were skeptics who doubted that the technology would ever scale. “Obviously,” they said, “once the machine has enough vacuum tubes some of them will fail and that will cause the entire computer to fail.” John von Neumann was annoyed by this line of reasoning, so he went off and proved a result we now know as **The Classical Fault-Tolerance Theorem**. Von Neumann showed given logic gates (like AND, OR, and NOT) which fail independently with some probability ϵ so long as ϵ is sufficiently small it’s possible to build a reliable circuit for any Boolean function f . Moreover, the circuit only needs to be a small factor larger than a circuit for f built of perfect gates. Informally this means you can build a reliable computer out of unreliable parts.

Admittedly, if the output of the circuit is just a single bit then that bit can’t possibly take the correct value with probability greater than $1 - \epsilon$ —for what if the very last gate is erroneous? However, one way to deal with this issue is to let the output be a long string of bits which is then processed by (say) a simple majority to get the value of f . This final majority computation is assumed to be error-free.

*How did von Neumann prove the classical fault-tolerance theorem? We won’t go through all the details, but the basic idea is to use the 3-bit repetition code recursively. Doing so pushes the probability of an error down further and further by taking a majority of majorities (on 9 bits for two levels of recursion), or a majority of majorities of majorities (on 27 bits for three levels), and so on. Now, this seems to raise a conceptual puzzle: *our error-correction circuits will**

themselves be subject to error, so when we compute these recursive majorities won't we simply be introducing more errors than we correct? Fortunately, von Neumann found that as long as the physical error probability ϵ is small enough each round of error-correction will be a “net win,” making things better rather than worse.

Anyway, von Neumann's whole idea ended up being mostly unnecessary when transistors replaced vacuum tubes, since transistors err with such minuscule probabilities (in your entire computer, with its billions of transistors, perhaps one transistor will output a wrong result per year when it's hit by a cosmic ray).

Could such a thing happen with quantum computing? That is, a “QC transistor,” which implements 1 and 2-qubit gates so reliably that error correction is then superfluous? Maybe! In the final lecture we'll discuss an approach to quantum computing called *topological quantum computing* which some physicists think could get us part of the way towards this dream. For now, though, we seem to be stuck in von Neumann's situation with quantum gates that *do* have significant probabilities of error.

Since we discussed D-Wave in the last lecture: a potentially fateful decision that D-Wave made early on was that they weren't going to do any error correction. However, even D-Wave might now be coming around to the view that some degree of error correction is necessary.

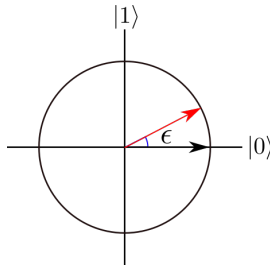
Crucially, *until* you get over the hurdle of error correction, it may not look like your quantum computer is doing much of anything useful. This is the main reason why progress in experimental quantum computing has often seemed slow (with the world record for Shor's algorithm remaining the factorization of 21 into 3×7 , etc...). Many people believe that practically important speedups will come only after we've overcome this hurdle.

27.2 Quantum Error Correction

You might worry that nothing like the 3-bit repetition code could possibly work in the quantum case, because quantum errors form a continuum; it's not a yes-or-no question whether an error has happened. For example, we might get the error

$$|0\rangle \rightarrow \sqrt{1 - \epsilon^2} |0\rangle + \epsilon |1\rangle. \quad (27.1)$$

It's said, only half-jokingly, that "80% of the work of building a quantum computer is reliably implementing the identity transformation."



Early in this course, though, we saw that the Quantum Zeno Effect (see Section 4.3) was a way to keep a drifting qubit in check. The trick is to just to keep measuring the qubit in the $|1\rangle, |1\rangle\}$ basis. If you get $|1\rangle$ then correct it to $|0\rangle$. If you get $|0\rangle$ then even if there had been an error now there's not!

It's like the joke where someone calls 911 to report a dead body and the operator asks the caller if they're sure the person is actually dead. Gunshots are heard; then the caller says, "OK, now what?"

But the quantum Zeno effect only solves the problem of quantum error-correction if the following two things are true:

- ▶ The only thing we're worried about is continuous drift (rather than, e.g., discrete bit-flips).
- ▶ We know a basis in which our qubit is supposed to be one of the basis vectors.

So how can we go further? If we only needed to correct bit-flip errors we could just use a quantum analogue of the 3-bit repetition code. Namely, we introduce the codewords (now quantum states instead of bit strings)

$$|\bar{0}\rangle = |000\rangle \quad \text{and} \quad |\bar{1}\rangle = |111\rangle \quad (27.2)$$

But bit-flip errors aren't the only things we're worried about! For an example of what else could go wrong let's look at the encodings of $|+\rangle$ and $|-\rangle$ under the above 3-bit repetition code:

$$|\bar{+}\rangle \rightarrow \frac{|000\rangle + |111\rangle}{\sqrt{2}} \quad \text{and} \quad |\bar{-}\rangle \rightarrow \frac{|000\rangle - |111\rangle}{\sqrt{2}} \quad (27.3)$$

How many qubits of $|\bar{+}\rangle$ do we need to act on to convert it to $|\bar{-}\rangle$? Only one! It suffices to apply a Z gate to any qubit. We call such an error a *phase-flip* error. You might think the problem is even worse still. Suppose we did manage to design a code that could correct both bit-flip errors and phase-flip errors. Even then, *aren't there infinitely many other ways for a qubit to err?*

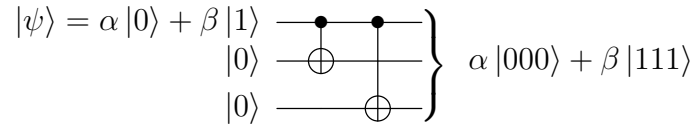


Figure 27.2: Circuit for encoding an input state in the 3-qubit bit-flip code.

Won't we need to add an infinite amount of additional redundancy to our code to account for all these different error modes?

Here's where a little piece of magic comes in and saves us. It turns out that if a quantum error-correcting code protects against both bit-flip and phase-flip errors then that's automatically enough to protect against all possible 1-qubit errors. *Why?* Without loss of generality let's assume that we have an error on the first qubit of the entangled state $|\psi_0\rangle$

$$|\psi_0\rangle = \alpha|0\rangle|v\rangle + \beta|1\rangle|w\rangle \quad (27.4)$$

A bit-flip error on the first qubit would result in

$$|\psi_1\rangle = \alpha|1\rangle|v\rangle + \beta|0\rangle|w\rangle \quad (27.5)$$

A phase-flip on the first qubit would result in

$$|\psi_2\rangle = \alpha|0\rangle|v\rangle - \beta|1\rangle|w\rangle \quad (27.6)$$

And both a bit-flip and a phase flip error (we'll call it a bit-phase flip for short) would result in

$$|\psi_3\rangle = \alpha|1\rangle|v\rangle - \beta|0\rangle|w\rangle \quad (27.7)$$

The key observation is that these four states constitute a basis for the 4-dimensional subspace of all possible states that you could reach from $|\psi_0\rangle$ by transformations on the first qubit. By linearity, this means that any linear transformation (be it unitary or even non-unitary) that you apply to the first qubit can be expanded as a superposition of bit-flip, phase-flip and bit-phase flip errors occurring to our qubit.

This is extremely similar to superdense coding. In both cases entanglement doubles the number of independent transformations that we can apply to a given qubit, while leaving a perfect record of which transformation was applied.

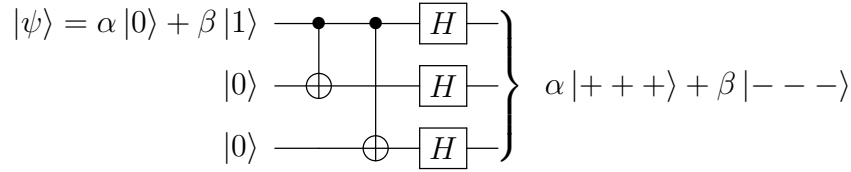


Figure 27.3: Circuit for encoding an input state into the 3-qubit phase-flip code.

So, the simplest possible goal of quantum error-correction is now as follows: assuming that the error was on the first qubit only get us back to $|\psi_0\rangle$ from wherever we might happen to be in this 4-dimensional “error subspace.” Simply measuring all the qubits would be a really bad idea—since even if that told us where we were in the error subspace it would be a “pyrrhic victory” that destroyed our quantum state in the process. Instead, maybe we can do a measurement that causes our state to collapse to one of the four basis vectors we saw $\{|\psi_0\rangle, |\psi_1\rangle, |\psi_2\rangle, |\psi_3\rangle\}$. Ideally this measurement should also return a flag indicating which of the states we collapsed to so that we can do the appropriate correction operation. Before we explain how to perform such a measurement, known as *syndrome detection*, we first need a code which can detect and correct against both bit-flip, phase-flip and bit-phase flip errors. For starters, if we just wanted to detect and correct just phase-flip errors we could do so using a code which is effectively the 3-qubit repetition code except in the Hadamard basis

$$|\bar{+}\rangle = |+\rangle|+\rangle|+\rangle \quad \text{and} \quad |\bar{-}\rangle = |-\rangle|-\rangle|-\rangle \quad (27.8)$$

OK, but just like the 3-qubit bit-flip code failed to protect against phase-flip errors, this code fails to protect against bit-flip errors. We can see this by first writing the logical states $|\bar{0}\rangle$ and $|\bar{1}\rangle$ using the above code

$$\begin{aligned} |\bar{0}\rangle &= \frac{1}{\sqrt{2}} (|\bar{+}\rangle + |\bar{-}\rangle) = \frac{1}{\sqrt{2}} (|+\rangle|+\rangle|+\rangle + |-\rangle|-\rangle|-\rangle) \\ &= \frac{1}{2} (|000\rangle + |011\rangle + |101\rangle + |110\rangle) \\ |\bar{1}\rangle &= \frac{1}{\sqrt{2}} (|\bar{+}\rangle - |\bar{-}\rangle) = \frac{1}{\sqrt{2}} (|+\rangle|+\rangle|+\rangle - |-\rangle|-\rangle|-\rangle) \\ &= \frac{1}{2} (|001\rangle + |010\rangle + |100\rangle + |111\rangle) \end{aligned} \quad (27.9)$$

We now observe that by applying a bit-flip to any qubit we can change $|\bar{0}\rangle$

to $|\bar{1}\rangle$ or vice versa. This observation brings us to our first serious quantum error-correcting code.

27.2.1 The Shor 9-Qubit Code

In 1995 Peter Shor proposed the first quantum error correction code able to correct against arbitrary single-qubit errors. His proposal was, simply put, to combine the bit-flip and phase-flip codes we've seen. For the first level of encoding, we're going to ensure protection from phase flip errors by encoding our logical $|\bar{0}\rangle$ and $|\bar{1}\rangle$ states using the 3-qubit phase flip code:

$$|\bar{0}\rangle = |+\rangle |+\rangle |+\rangle \quad \text{and} \quad |\bar{1}\rangle = |-\rangle |-\rangle |-\rangle. \quad (27.10)$$

Note we're labeling our codeword differently than we did in Equation 27.8 (that's ok, we have some freedom in how we choose to label the codewords). As we noted above though, this encoding is susceptible to bit-flip errors. To protect from bit-flip errors we take each of the states comprising the codewords above and encode those using the 3-qubit bit-flip code, which yields:

$$|\bar{0}\rangle = \left(\frac{|000\rangle + |111\rangle}{\sqrt{2}} \right)^{\otimes 3} \quad \text{and} \quad |\bar{1}\rangle = \left(\frac{|000\rangle - |111\rangle}{\sqrt{2}} \right)^{\otimes 3} \quad (27.11)$$

We claim that this code lets us detect and correct a bit-flip or a phase-flip (and hence, any possible error) on any one of the 9 qubits.

How does this detect and correct bit-flip errors? We just need build a quantum circuit that checks whether all 3 of the qubits in a given block have the same value and if they don't it sets the wayward qubit equal to the majority of all 3 qubits in the block. We then apply that circuit to each of the 3 blocks separately. The circuit for performing this check can be found in Figure 27.5.

More interestingly, *how does this code also detect and correct phase-flip errors?* We can build a quantum circuit that computes the relative phase between $|000\rangle$ and $|111\rangle$ within each block, checks whether all 3 phases have the same value, and sets any wayward phases equal to the majority of all 3 phases. A circuit for performing this check can be found in Figure 27.6.

We can combine both of the error checking circuits in Figures 27.5 and 27.6. We call the resulting circuit the *syndrome detecting circuit*. We first apply the circuit from 27.5 to each of the three blocks of qubits encoded with the bit-flip code in order to fix any bit-flip errors in these blocks. We then apply a version of 27.6 to detect phase-flip errors on a block-by-block basis. Note that a phase-flip on any qubit in a block modifies the overall state of the

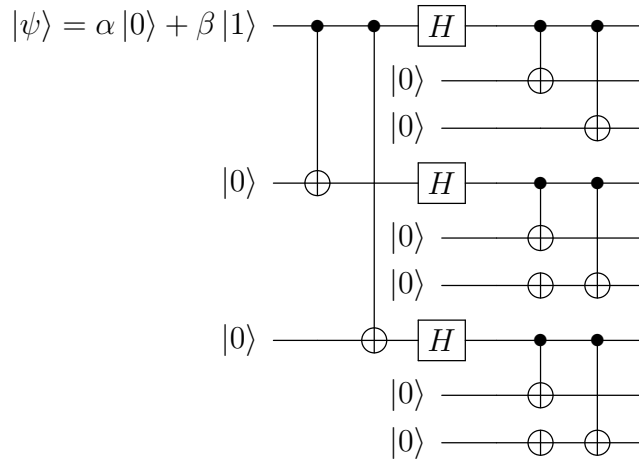


Figure 27.4: Encoding circuit for Shor's 9-qubit code. Indentation meant to emphasize the concatenated nature of the encoding scheme. Reproduced from Nielsen and Chuang's "Quantum Computation and Quantum Information."

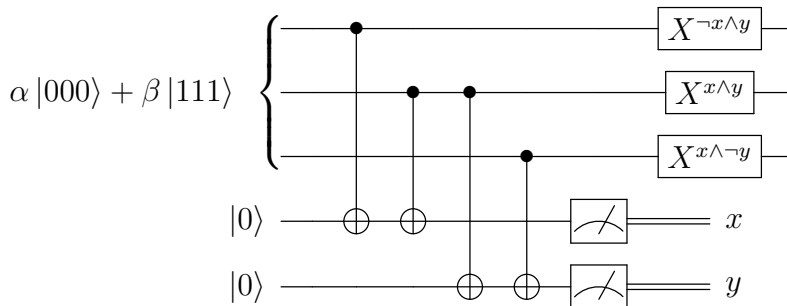


Figure 27.5: Circuit for detecting and correcting bit-flip errors on an encoded input state. The final round of X gates is applied conditionally based on the values of x and y observed.

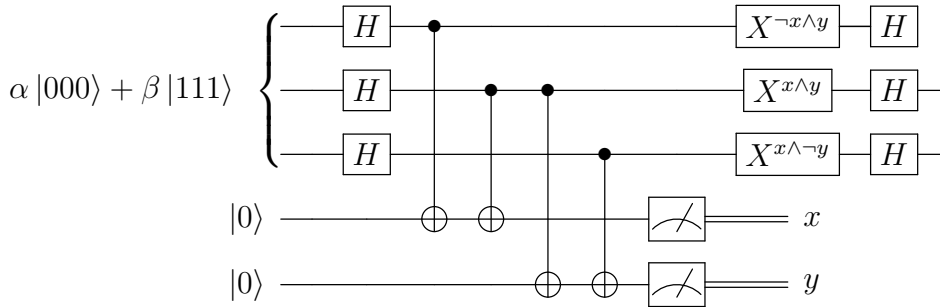


Figure 27.6: Circuit for detecting and correcting phase-flip errors on an encoded input state. The final round of X gates are applied conditionally based on the values of x and y observed.

qubits in the block in exactly the same way (you can verify this by applying a phase-flip error to the phase-flip code codewords and seeing what happens). The final measurements applied to the ancillary qubits in Figure 27.7 produce an output string whose value depends on precisely which error occurred and on which qubit; this output string is called the *syndrome*.

Table 27.1: Syndrome-error correspondence for the bit-flip code. Subscripts on the errors denote which qubit the error occurred on.

x	y	Error
0	0	No Error
1	0	X_0
1	1	X_1
0	1	X_2

The complete table of possible syndromes and the corresponding errors is too long to warrant including here, but in Tables 27.1 and 27.2 we give this table for the bit-flip and phase-flip codes respectively—since the Shor code is a combination of these two codes, the syndrome-error correspondence follows from these two tables in a fairly straightforward way. One of the key requirements is that there be a one-to-one correspondence between the syndromes and the errors (otherwise we couldn't reliably reverse the errors), which we can see from Tables 27.1 and 27.2 is satisfied here.

The measurements we apply to the ancilla not only have the effect of giving us information about which error occurred, they also have the effect of

Table 27.2: Syndrome-error correspondence for the phase-flip code. Subscripts on the errors denote which qubit the error occurred on.

x	y	Error
0	0	No Error
1	0	Z_0
1	1	Z_1
0	1	Z_2

collapsing our state such that *only* the error corresponding to the syndrome occurred—subject to the assumption that we really only had one qubit experience an error. We recommend trying this out directly for an arbitrary logically encoded input state and a few different errors. The reason the state collapses following the measurement is that our syndrome detection circuit generates entanglement between the qubits we use to encode our state and the ancilla qubits we use to detect the errors.

*There is no small irony in the fact that in our battle against decoherence—resulting from unwanted entanglement between our system and the environment—one of our main weapons is the strategic use of **more** entanglement!*

You can check that both of the operations above work not just for $|\bar{0}\rangle$ and $|\bar{1}\rangle$, but for arbitrary superpositions of the form $\alpha|\bar{0}\rangle + \beta|\bar{1}\rangle$. So, the above will fix any stray unitary transformations that get applied to any one qubit.

But what about errors that involve decoherence or measurement? We claim that once we've handled all possible unitary transformations, we've automatically handled all possible errors—because an arbitrary error might turn pure states into mixed states, but it still keeps us within the same 4-dimensional subspace. The measurements performed at the end of the circuits in Figure 27.7 still project us down to one of the orthogonal states corresponding to some particular error occurring. We can still get back to our original state $|\psi_0\rangle$ by applying bit flips and phase flips as needed.

Shor's 9-qubit code was the first quantum error correcting code. Not long afterward Andrew Steane found a shorter code that could also detect and correct any error on 1 qubit. Steane's code encoded 1 logical qubit into only 7 physical qubits. Then, Raymond Laflamme and others found codes that used only 5 qubits. Five qubits turns out to be the least possible if you want both to detect and to correct an arbitrary 1-qubit error, just like 3 bits is the least

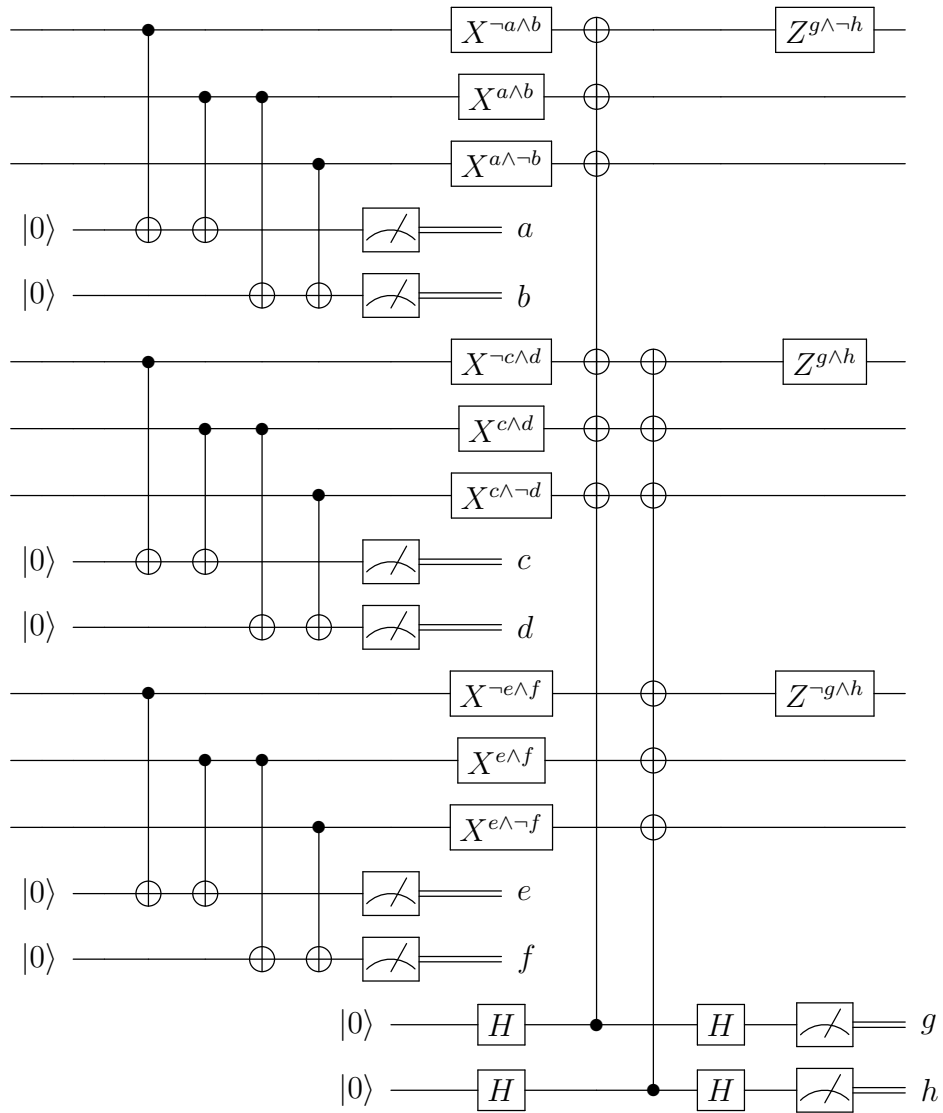


Figure 27.7: Quantum circuit for performing the full syndrome detection and error correction procedure for the Shor 9-qubit code. We’ve structured the circuit to highlight the concatenated nature of the error correction. We first use the circuit from 27.5 on each block of 3 qubits to detect and correct any bit-flip errors. We then use a version of the syndrome detection circuit in 27.6—we’ve made liberal use of the identities $X = HZH$ and $(H \otimes H)(\text{CNOT}_{i \rightarrow j})(H \otimes H) = \text{CNOT}_{j \rightarrow i}$, where the subscripts on $\text{CNOT}_{i \rightarrow j}$ denote that qubit i controls qubit j , to reduce the size of the circuit for the sake of brevity—to detect and correct any phase-flip errors.

possible for a classical error correcting code. We won't prove this.

In the next lecture we'll discuss the *stabilizer formalism*, which gives us an amazingly compact and efficient notation for manipulating these sorts of quantum error-correcting codes.

27.2.2 Quantum Fault Tolerance

So, we can encode a qubit, let the qubit sit around passively and protect it against a single physical error. *How about doing a full fault tolerant quantum computation?* That's the subject of the famous **Quantum Fault-Tolerance Theorem**, also known as the **Threshold Theorem**. Proved independently by several groups (Aharonov & Ben-Or / Zurek et al. ~ 1996), the theorem says the following: suppose that in your quantum computer, each qubit fails at each time step with independent probability ϵ (where "fails" could mean, e.g., it is replaced by the maximally mixed state). Then assuming we're able to:

- ▶ Apply gates to many qubits in parallel.
- ▶ Measure and discard bad qubits.
- ▶ Pump in fresh $|0\rangle$ qubits
- ▶ Do extremely fast and reliable classical computation.

we can *still* solve any problem in **BQP**, so long as ϵ is sufficiently small. Moreover, we can simulate any error-free quantum circuit with only an additional $\text{polylog}(n)$ -factor overhead in the number of gates.

The initial estimates for ϵ were $\sim 10^{-6}$. That's since been improved by 3-4 orders of magnitude, depending on the assumptions you're willing to make.

Since its discovery, the threshold theorem has set much of the research agenda for experimental quantum computing. It says that once we can decrease error below a certain threshold, we'll effectively be able to make it arbitrarily small by applying multiple recursive layers of quantum error-correction. Journalists often try to gauge progress in experimental quantum computing by asking about the number of qubits, but at least as important is the reliability of the qubits. It's reliability that will determine when (if ever) we cross the threshold that would let us get to arbitrarily small error and add as many additional qubits as we liked.

No one there yet, but lots of progress is being made on two fronts:

- ▶ Making physical qubits more reliable
 - In the earliest quantum computing experiments, the probability of failure per qubit per time step was of order ~ 1 , meaning that the quantum state would barely remain coherent at all. But over the last twenty years, there have been improvements in reliability by several orders of magnitude (in multiple proposed architectures including superconducting qubits and trapped ions). Today, anyone can access a 5-qubit superconducting device over the Internet (IBM’s “Quantum Experience”) that has decoherence rates that weren’t been achievable a decade ago. Meanwhile, a few years ago the group of John Martinis at Google demonstrated decoherence rates for one or two qubits in isolation that are already below the current fault-tolerance thresholds. *So is the problem solved?* Unfortunately not, because integrating more qubits on a single chip pushes the decoherence rate back up. The challenge now is to figure out how to scale up to 100 or 300 qubits while still keeping the error rate down.
- ▶ Inventing better error-correcting codes and fault-tolerance schemes
 - There are many tradeoffs here. For example, it’s now known (at least heuristically) how to handle error rates of up to 3-5%, but only via fault-tolerance schemes that use thousands of physical qubits for every one logical qubit. One big challenge is to invent schemes that can handle large error and *also* have low overheads. Topological qubits, could also help with this tradeoff.

Here’s one milestone that has been achieved fairly recently, in 2016 the research groups of Michel Devoret and Robert Schoelkopf at Yale reported the use of a quantum error correcting code to keep a logical qubit alive for longer than the physical qubits comprising it.

Lecture 28: The Stabilizer Formalism

In this lecture we'll see a beautiful formalism that was originally invented to describe quantum-error correcting codes, but now plays many different roles in quantum computation. First, some definitions:

- ▶ **Stabilizer Gates** are the gates CNOT, Hadamard and $S = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$ (also called the “Phase Gate”).
- ▶ **Stabilizer Circuits** are quantum circuits made entirely of stabilizer gates.
- ▶ **Stabilizer States** are the states that a stabilizer circuit can generate starting from $|0 \cdots 0\rangle$.

We briefly met stabilizer gates earlier in the course when we discussed universal quantum gate sets and needed to include a warning that the set $\mathcal{S} = \{\text{CNOT}, H, S\}$ is not universal. This failure of universality might be surprising. After all, the set \mathcal{S} seems to have everything we need: the Hadamard gate can create superpositions, CNOT acts on two qubits and can create complicated entangled states and S can even add complex phases. What's more, many of the weird quantum effects and protocols that we saw in this course can be demonstrated entirely using stabilizer gates. Examples include: superdense coding, quantum teleportation, BB84 quantum key distribution, Wiesner's quantum money, the Deutsch-Jozsa algorithm, the Bernstein-Vazirani algorithm and the Shor 9-qubit code.

So then, *what prevents \mathcal{S} from being universal?* Well, if you try playing around with the CNOT, Hadamard and S gates you'll notice that you tend to reach certain discrete states, but never anything between them. You'll also notice that whenever you create an n -qubit superposition that assigns nonzero amplitudes to the strings in some set $A \subseteq \{0, 1\}^n$, it's always an

equal superposition over A (possibly with ± 1 or $\pm i$ phases). Furthermore, A is always an affine subspace of F_2^n (so in particular, $|A|$ is always a power of 2).

With only 1 qubit, the H and S gates can only get us to 6 states in total (ignoring global phases), via the reachability diagram shown on the right. These 6 states, $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle, |i\rangle, |-i\rangle\}$, are the 1-qubit stabilizer states.

What about with two qubits? Now you can reach some more interesting states, like $\frac{|00\rangle + i|11\rangle}{\sqrt{2}}$ or $\frac{|01\rangle - i|10\rangle}{\sqrt{2}}$. But these always follow certain patterns as mentioned above. For example, they're always equal superpositions over a numbers of strings that is a power of 2. Moreover, a measurement of a given qubit in the $\{|0\rangle, |1\rangle\}$ basis always produces one of the possible outcomes:

- ▶ $|0\rangle$ with probability 1
- ▶ $|1\rangle$ with probability 1
- ▶ $|0\rangle$ and $|1\rangle$ with equal probabilities

So what gives? In order to explain, it will help to define a few concepts. We say that a unitary U stabilizes a pure state $|\psi\rangle$ if $U|\psi\rangle = |\psi\rangle$. In other words, if $|\psi\rangle$ is an eigenstate of U with eigenvalue $+1$. Crucially, *global phase matters here!* If $U|\psi\rangle = -|\psi\rangle$, then U does *not* stabilize $|\psi\rangle$.

Notice that if U and V both stabilize $|\psi\rangle$, then any product of them, like UV or VU , also stabilizes $|\psi\rangle$. Likewise, if U stabilizes $|\psi\rangle$, then its inverse U^{-1} must also stabilize $|\psi\rangle$. The identity matrix stabilizes everything. Since matrix multiplication is associative, this means that the set of unitaries that stabilize $|\psi\rangle$ forms a *group* under multiplication.

The next ingredient we need is the **Pauli Matrices**. We've already encountered these four matrices, and they come up constantly in quantum physics. To remind you, they are:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad (28.1)$$

Notice that these matrices match up with the errors we needed to worry about in quantum error-correction (the last one up to a global phase).

$$\text{No Error} \rightarrow I \quad \text{Bit-flip} \rightarrow X \quad \text{Phase-flip} \rightarrow Z \quad \text{Bit-flip + Phase-flip} \rightarrow Y$$

Note that for Y gate this is equivalent to the bit-phase-flip error we saw in the previous lecture up to a global phase. The Pauli matrices satisfy several

beautiful identities:

$$\begin{aligned}
 XY &= iZ & YX &= -iZ \\
 YZ &= iX & ZY &= -iX \\
 ZX &= iY & XZ &= -iY \\
 X^2 &= Y^2 = Z^2 = I
 \end{aligned} \tag{28.2}$$

If you've seen the quaternions, you might recall that they're defined using the same kinds of relations. This is not a coincidence!

Nothing is a coincidence in math!

In addition, all four Pauli matrices are both unitary and Hermitian. *So what does each Pauli matrix stabilize?*

$$\begin{array}{ll}
 I \text{ stabilizes everything} & -I \text{ stabilizes nothing} \\
 X \text{ stabilizes } |+\rangle & -X \text{ stabilizes } |-\rangle \\
 Z \text{ stabilizes } |0\rangle & -Z \text{ stabilizes } |1\rangle \\
 Y \text{ stabilizes } |i\rangle & -Y \text{ stabilizes } |-i\rangle
 \end{array}$$

So each of the six 1-qubit stabilizer states has a corresponding Pauli matrix that stabilizes it.

Next, given an n -qubit pure state $|\psi\rangle$, we define $|\psi\rangle$'s stabilizer group to be the group of all tensor products of Pauli matrices that stabilize $|\psi\rangle$. We know this is a group since the set of Pauli matrices is closed under multiplication and so is stabilizing $|\psi\rangle$. Stabilizer groups have the additional property of being Abelian (the group multiplication operation is commutative).

To illustrate, the stabilizer group of $|0\rangle$ is $\{I, Z\}$. The stabilizer group of $|+\rangle$ is $\{I, X\}$. The stabilizer group of $|0\rangle \otimes |+\rangle$ is the Cartesian product of those two groups:

$$\{I \otimes I, I \otimes X, Z \otimes I, Z \otimes X\}. \tag{28.3}$$

As a convention, we'll omit the \otimes 's from now on when unambiguous. For the example above this convention gives us $\{II, IX, ZI, ZX\}$.

For a slightly more interesting example, *what's the stabilizer group of a Bell pair?* We know XX is in it because

$$\frac{(X \otimes X) |00\rangle + (X \otimes X) |11\rangle}{\sqrt{2}} = \frac{|11\rangle + |00\rangle}{\sqrt{2}} = \frac{|00\rangle + |11\rangle}{\sqrt{2}}.$$

By a similar argument, $-YY$ must be in the group. We can get another element by multiplying these two elements (remember the product of two stabilizers is a stabilizer)

$$(XX)(-YY) = (X \otimes X)(-Y \otimes Y) = -(iZ) \otimes (iZ) = Z \otimes Z = ZZ$$

So the stabilizer group of $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$ is $\{II, XX, -YY, ZZ\}$. You can check that it doesn't contain anything else. You can similarly compute the stabilizer group of $\frac{|00\rangle-|11\rangle}{\sqrt{2}}$ to be $\{II, -XX, YY, ZZ\}$.

Now, here's an amazing fact, which we won't prove: the n -qubit stabilizer states are exactly the n -qubit states that have a stabilizer group of size 2^n . So the 1-qubit stabilizer states are those states with a 2-element stabilizer group, the 2-qubit stabilizer states are those states with a 4-element stabilizer group and so on. This is a completely different characterization of the stabilizer states, which gives deeper insight into their structure. It makes no mention of stabilizer circuits, but tells us something about the *invariant* that stabilizer circuits are preserving.

OK, so suppose we have an n -qubit stabilizer state which (by the above) has a 2^n -element stabilizer group G . Then here's the next thing we might want to know: *How can we succinctly specify G ?* In particular we want to know whether G always has a small generating set—that is, a few elements from which we can get all the others by multiplication. The answer turns out to be yes—indeed, *every* finite group G has a generating set with at most $\log_2(|G|)$ elements (proof left as an exercise).

More concretely, given any n -qubit stabilizer state, its stabilizer group is always generated by only n elements (i.e., signed tensor products of Pauli matrices). So, to specify a stabilizer group (and hence, a stabilizer state) you only need to specify n generators. Let's see an example. To specify the Bell pair, which has stabilizer group $\{II, XX, -YY, ZZ\}$, it's enough to give the following generating set:

$$\begin{bmatrix} XX \\ ZZ \end{bmatrix}$$

We could also give a different generating set, like

$$\begin{bmatrix} XX \\ -YY \end{bmatrix}$$

How many bits does it take to store such a generating set in your computer? Well, there are n generators and each one takes $2n + 1$ bits to specify: 2 bits

for each of the n Pauli matrices plus 1 additional bit for the sign. So the total number of bits is $n(2n + 1) = 2n^2 + n = \mathcal{O}(n^2)$. Naïvely writing out the entire amplitude vector or the entire stabilizer group would have taken $\sim 2^n$ bits, so we've gotten an exponential savings. We're already starting to see the power of the stabilizer formalism.

28.1 The Gottesman-Knill Theorem

But the power of the stabilizer formalism turns out to go much further. Around 1998, Daniel Gottesman and Manny Knill proved the **Gottesman-Knill Theorem**. This theorem says that there's a polynomial-time classical algorithm to simulate any stabilizer circuit that acts on a stabilizer state (e.g. $|0 \cdots 0\rangle$). Here "simulate" means pretty much anything you could ask for: you can compute the probability of any possible sequence of measurement outcomes or you can simulate the measurement outcomes if given access to a random bit source.

A pessimistic interpretation of this theorem is that stabilizer states and gates, by themselves, are useless for producing superpolynomial quantum speedups.

So, *how does the classical simulation work?* In a nutshell it works by keeping track at each point in time of a list of generators for the current state's stabilizer group. The algorithm then updates the list of generators whenever a CNOT, Hadamard, Phase, or measurement gate is applied.

Almost the only time that Professor Aaronson (being a theorist) ever wrote code that other people actually used was when he did a project in grad school for a computer architecture course. He wrote a fast simulator for stabilizer circuits called CHP which could handle thousands of qubits on a normal laptop (limited only by the available RAM). The challenge of actually implementing the Gottesman-Knill algorithm in an optimized way led to the discovery of an even faster classical algorithm for simulating stabilizer circuits, so Aaronson ended up publishing a paper with Gottesman about this. Truth be told, this project had very little to do with computer architecture; he's still not sure why the professor accepted it!

28.1.1 The Gottesman-Knill Algorithm

So how does the Gottesman-Knill algorithm work? For simplicity, let's assume the initial state is $|0 \cdots 0\rangle$. Then the first step is to find a stabilizer representation (that is, a list of generators) for $|0 \cdots 0\rangle$. We know the stabilizer group contains $I \cdots I$, but we won't put that into the generating set (it's implied). Since $|0\rangle$ is a $+1$ eigenstate of Z you can check that the following generating set works:

$$\begin{bmatrix} ZIII \cdots I \\ IZII \cdots I \\ IIZI \cdots I \\ \vdots \\ IIII \cdots Z \end{bmatrix}$$

For purposes of the algorithm it's useful to write these lists of generators in a slightly different way which we call the **Tableau Representation**. Here we'll keep track of two $n \times n$ matrices of 1's and 0's (as well as n signs). The two matrices can be combined entrywise to represent an $\{I, X, Y, Z\}$ matrix like the one above. For the state $|0000\rangle$ the tableau representation is given by the following:

$$\begin{aligned} &+ \left[\begin{array}{cccc|cccc} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right] \end{aligned} \quad (28.4)$$

In the tableau above, each row represents a particular generator of the stabilizer group. The first column gives the overall sign of each generator. Of the remaining columns, the first half of the columns (the second through fifth columns above) are the “ X -matrix” and the second half of the columns are the “ Z -matrix.” The values in the i^{th} position of some row of the X and Z matrices specify the corresponding Pauli matrix in the i^{th} position of the generator's tensor product. A 0 in both the X and Z -matrices represents an identity in the generator. A 1 in the X -matrix and 0 in the Z -matrix or vice versa represents either an X or a Z respectively. Finally, a 1 in both the X and Z matrices represents a Y in the generator. Thus, the first row of the above tableau represents the generator $+ZIII$, the second $+IZII$, the third $+IIZI$ and the fourth $+IIIZ$. So this is just another way to represent the generating set $\{ZIII, IZII, IIZI, IIIZ\}$ for the state $|0000\rangle$.

We're now going to provide rules for updating this tableau representation whenever a CNOT, Hadamard, or Phase gate is applied. We won't prove that

the rules are correct, but you should examine them one by one and see if you can convince yourself. We're also going to cheat a little. Keeping track of the $+$'s and $-$'s is tricky and not particularly illuminating, so we'll just ignore them. *What do we lose by ignoring them?* Well, whenever measuring a qubit has a definite outcome (say, either $|0\rangle$ or $|1\rangle$), we need the $+$'s and $-$'s to figure out which of the two it is. On the other hand, if we only want to know whether measuring a qubit will give a definite outcome or a random outcome (and not which definite outcome in the former case), then we can ignore the signs.

The Algorithm:

- ▶ To apply H to the i^{th} qubit:
 - Swap the i^{th} column of the X -matrix and the i^{th} of the Z -matrix.

This should be pretty intuitive: the whole point of the Hadamard gate is to “swap the X and Z bases.”

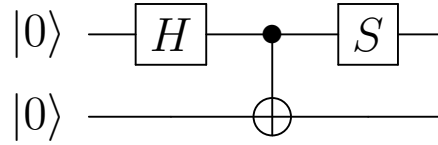
- ▶ To apply S to the i^{th} qubit:
 - Bitwise XOR the i^{th} column of the X -matrix into the i^{th} column of the Z -matrix.
- ▶ To apply CNOT from with the i^{th} qubit as the control and the j^{th} as the target:
 - Bitwise XOR the i^{th} column of the X -matrix into the j^{th} column of the X -matrix.
 - Bitwise XOR the j^{th} column of the Z -matrix into the i^{th} column of the Z -matrix.

The first of these rules seems reasonable enough. For the second rule, we need to remember that a CNOT from i to j is equivalent to a CNOT from j to i when viewed in the Hadamard basis!

- ▶ Finally, whenever the i^{th} qubit is measured in the $\{|0\rangle, |1\rangle\}$ basis the measurement will have a determinate outcome if and only if the i^{th} column of the X matrix is all 0's.

There are also rules for updating the tableau in case the measurement outcome is not determinate, but we won't cover them here.

Here's another cool fact: the number of basis states that have nonzero amplitudes is just 2^k , where k is the rank of the X -matrix. In the example in Equation 28.4, $\text{rank}(X) = 0$, corresponding to the fact that our “superposition” only contains a single basis state, namely $|0000\rangle$.



Let's test this all out by keeping track of the tableau for the circuit above. We start with the state $|00\rangle$ which has the tableau representation

$$\left[\begin{array}{cc|cc} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array} \right]. \quad (28.5)$$

Applying the Hadamard to the first qubit has effect of swapping the first columns of the X -matrix and Z -matrix.

$$\left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{array} \right] \quad (28.6)$$

You could convert this back into the generators by saying that the current state is the one generated by $+XI$ and $+IZ$; this makes sense since those do indeed generate the stabilizer group for $|+\rangle|0\rangle$. Now, to apply the CNOT we bitwise XOR the first column in the X -matrix into the second column in the X -matrix and likewise bitwise XOR the second column in the Z -matrix into the first column in the Z -matrix. This results in the tableau

$$\left[\begin{array}{cc|cc} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{array} \right]. \quad (28.7)$$

The generators corresponding to this tableau are $\{XX, ZZ\}$, which as we saw earlier are indeed the stabilizer generators for a Bell pair as we expect. Finally, we apply the Phase gate by bitwise XORing the first column in the X -matrix into the first column of the Z -matrix. This results in the tableau

$$\left[\begin{array}{cc|cc} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{array} \right]. \quad (28.8)$$

This final tableau corresponds to the generators $\{YX, ZZ\}$, which you can check are the stabilizers for the state $\frac{|00\rangle+i|11\rangle}{\sqrt{2}}$.

28.2 Stabilizer Codes

A **Stabilizer Code** is a quantum error-correcting code in which the encoding and decoding can be done entirely by stabilizer circuits. In particular, this means that all the codewords are stabilizer states. In quantum computing

research, most (though not all) of the error-correcting codes that have been seriously considered are stabilizer codes. The reason is similar to why linear codes play such a central role in classical error correction. Namely, it makes everything much easier to calculate and reason about, and by insisting on it we don't seem to give up any of the error-correcting properties that we want. As a result, the stabilizer formalism is the lingua franca of quantum error-correction; it's completely indispensable in this setting.

Shor's 9-qubit code is an example of a stabilizer code. Recall that with that code we had the codewords $\frac{(|000\rangle \pm |111\rangle)^{\otimes 3}}{2\sqrt{2}}$. These two codewords correspond to the stabilizer group generators below.

$$\left[\begin{array}{cccccccccc} Z & Z & I & I & I & I & I & I & I & I \\ I & Z & Z & I & I & I & I & I & I & I \\ I & I & I & Z & Z & I & I & I & I & I \\ I & I & I & I & Z & Z & I & I & I & I \\ I & I & I & I & I & I & Z & Z & I & I \\ I & I & I & I & I & I & I & Z & Z & I \\ X & X & X & X & X & X & I & I & I & I \\ I & I & I & X & X & X & X & X & X & X \\ \pm & X & X & X & X & X & X & X & X & X \end{array} \right] \quad (28.9)$$

The sign on the last line gives either $|\bar{0}\rangle$ for $+$ or $|\bar{1}\rangle$ for $-$.

We can use intuition to see why the above elements are in the stabilizer group. Firstly, phase-flips applied to any pair of qubits in the same block cancel each other out. Secondly, bit-flips applied to all of the qubits in a given block also take us back to where we started, though possibly with the addition of a global -1 phase. You then just need to check that these 9 elements are linearly independent of each other, meaning that there aren't any more to be found.

Now that we know the stabilizer formalism, we're finally ready to see an "optimal" 5-qubit code for detecting and correcting an arbitrary error on any one qubit. The codeword states would be a mess if we wrote them out explicitly—they're given by superpositions over 32 different 5-bit strings! Everything is much more compact if we use the stabilizer formalism. The code corresponds to the following set of stabilizer group generators:

$$\left[\begin{array}{ccccc} X & Z & Z & X & I \\ I & X & Z & Z & X \\ X & I & X & Z & Z \\ Z & X & I & X & Z \\ \pm & X & X & X & X \end{array} \right] \quad (28.10)$$

Once again the sign on the last generator is $+$ if we want the $|\bar{0}\rangle$ state or $-$ if we want the $|\bar{1}\rangle$ state. One can check (we won't prove it here) that this code can indeed detect and correct a bit-flip, phase-flip, or bit-phase-flip error on any one of the five qubits.

28.2.1 Transversal Gates

To conclude this lecture let's say a tiny bit about doing actual quantum computations on qubits that are encoded using stabilizer codes. Suppose we have n logical qubits, each encoded with a stabilizer code, and we want to apply a gate to one or two of the logical qubits. The "obvious" way to do this would be:

- ▶ First decode the qubits.
- ▶ Apply the desired gate to the "bare," unencoded qubits.
- ▶ Re-encode the result.

But doing all that is expensive and creates lots of new opportunities for error! While the qubits are unencoded there's nothing to protect them from decoherence. So it would be awesome if we had a code where applying gates to encoded qubits was hardly more complicated than applying them to unencoded qubits. This motivates the following definition: the gate G is *transversal* for the code C if in order to apply G to logical qubits encoded using C , all you need to do is apply G independently to each of the physical qubits. For example, the Hadamard gate is transversal if you can Hadamard a logical qubit by just separately Hadamarding each of the physical qubits. You should check, for example, that the Hadamard gate is transversal for Shor's 9-qubit code.

It turns out that there are quantum error-correcting codes for which the CNOT, Hadamard, and Phase gates are all transversal. Thus, if you use one of these codes, then applying any stabilizer circuit to the encoded qubits is extremely cheap and easy. Unfortunately, we already saw that the stabilizer gates are non-universal. Moreover, there's a theorem due to Zeng, Cross and Chuang in 2007 that says that for any useful stabilizer code C the corresponding set of transversal gates can't be universal. Eastin and Knill generalized this to show that *no* useful quantum error-correcting code can have a universal set of transversal gates. This means that if we want a universal error-corrected quantum computer we're going to need to figure out how to implement some non-stabilizer gate (say Toffoli or T) in a non-transversal manner, probably via some sequence of gates that is much more expensive.

Quantum computing researchers who study this problem tend to adopt a worldview wherein stabilizer gates are “free”—they’re so cheap to implement that you might as well not even count them—and the “complexity” of a quantum circuit equals the number of non-stabilizer gates needed to implement it. The non-stabilizer gates are so much more expensive that they completely dominate the running time.

In practice, a lot of this research has boiled down to designing improved and less-expensive methods for getting non-stabilizer gates into a circuit. There are various tricks, a famous example being **Magic State Distillation**. The idea here is that if you can just produce certain non-stabilizer states like $\frac{|0\rangle + e^{i\pi/4}|1\rangle}{\sqrt{2}}$ (so-called “magic states”), then by applying certain stabilizer operations to those states and performing measurements (adapting your procedure as necessary based on the outcome of the measurements), you can simulate the effect of applying non-stabilizer gates. In Figure 28.1 we give a circuit which uses the state $\frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)$ along with stabilizer gates and measurements to simulate the effect of applying a T gate. In other words: with help from magic states, stabilizer operations can break out of the “Gottesman-Knill prison” and get all the way up to universal quantum computation. On the other hand, actually realizing this idea seems to require building a quantum computer where the overwhelming majority of the work would happen in “magic state factories,” with the actual quantum computation on the magic states almost an afterthought.

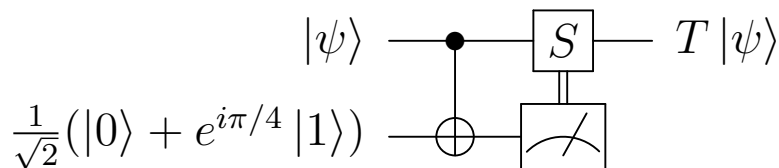


Figure 28.1: Circuit for simulating a T gate using stabilizer operations and measurements with the magic state $\frac{1}{\sqrt{2}}(|0\rangle + e^{i\pi/4}|1\rangle)$.

There’s a different way to understand the importance of non-stabilizer gates for quantum computation. The paper by Aaronson and Gottesman from 2004, mentioned earlier, also proved the following result: suppose we have a quantum circuit on n qubits, which contains mostly stabilizer gates (say, $n^{\mathcal{O}(1)}$ of them) but also a small number t of non-stabilizer gates. Then there’s a classical algorithm to simulate the circuit in time that’s polynomial in n and exponential in t . This tells us that if we want an exponential quantum speedup, then not only do we need non-stabilizer gates in our circuit, we need *many* such gates.