Quantum Complexity Theory

Lecture Notes

Daniel Grier

Contents

Preamble

These lecture notes derive from a sequence of scribe notes taken from the Fall 2022 iteration of CSE 291 / Math 277A (Quantum Complexity Theory). This document represents a continual and iterative process to bring those notes into a more cohesive whole. Any mistakes can be assumed to have been introduced by me. Please feel free to email me (`dgrier@ucsd.edu`) if you notice any.

These course notes are written for graduate students with a strong mathematical background, but not necessarily any previous experience with quantum computing. Some previous exposure to complexity theory will be extremely useful. I recommend the excellent Arora-Barak textbook for those looking to brush up on that background.

Overview

The goal of these notes is to rigorously compare, using the tools of complexity theory, the power of quantum and classical computers. We will see some settings in which quantum computers outperform their classical counterparts and some settings in which quantum computers are no better than brute force classical approaches. Given the recent explosion of progress in actually building a quantum computer, it is becoming more important than ever that we understand the difference in the quantum and classical worlds and what they allow us to compute. The exploration in these notes will take us to the forefront of quantum computing research, where we'll look at the complexity-theoretic foundations of these recent experiments.

Chapter 1

Foundations of Quantum Mechanics

Before we can reason about the power of quantum computers, we must obviously first understand what kinds of computations they unlock. We will start with the pure foundations: What is a quantum state, and what kinds of operations can you perform on that state?

From there, we will define a computational model (analogous to the classical Turing machine) that captures the essence of a quantum computer.

1.1 The basics of quantum computation

What is the state of a quantum system? Let's start by analogy to one of the simplest classical objects—a biased coin. Since it will be convenient later, let's suppose the coin has two sides, corresponding to a 0-outcome and a 1-outcome (perhaps more traditionally these two outcomes would be called "heads" and "tails").

To be even more concrete, let's suppose the coin is biased so that it lands on the 0-outcome with 30% probability and on the 1-outcome with 70% probability. Suppose we flip the coin in the air, and we want to describe the probability distribution over outcomes when the coin lands. We could represent it by the length-2 vector:

$$\begin{pmatrix} 0.3 \\ 0.7 \end{pmatrix} \begin{matrix} \leftarrow \text{Probability of } 0\text{-outcome} \\ \leftarrow \text{Probability of } 1\text{-outcome} \end{matrix}$$

In some sense, this represents the "state" of the coin if we know the coin has landed on one side or the other, but we have not yet looked at which outcome.

If we *were* to look at the outcome, then the state of the system immediately changes to whichever outcome we saw:

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \leftarrow \begin{matrix} 0\text{-outcome} \\ \text{with certainty} \end{matrix} \qquad \text{or} \qquad \begin{pmatrix} 0 \\ 1 \end{pmatrix} \leftarrow \begin{matrix} 1\text{-outcome} \\ \text{with certainty} \end{matrix}$$

since there is no ambiguity in the outcome once we've observed it.

Stepping back a bit, let's look at the full description of states and operations in this classical probability framework. First, notice that instead of a coin with just 2 outcomes, we could have as many outcomes as we like (think of a biased die); but for simplicity, let's assume there are only finitely many. In a system with d outcomes, the state of the system would be described

by a vector of d probabilities. The key property of this vector is that each probability is non-negative and all probabilities sum up to 1.

The set of operations that we could perform on this system are the set of operations that take probability vectors to probability vectors. Specifically (and we will see how this changes in the quantum setting soon), these operations preserve the ℓ_1-norm of the vector, where the ℓ_1-norm of vector $v = (v_1, v_2, \ldots, v_d) \in \mathbb{C}^d$ is defined as

$$\|v\|_1 := \sum_{i=1}^{d} |v_i|$$

Qubits

Let's now complete the analogy of the classical probabilistic bit discussed above with the quantum variant called a *qubit*. Instead of assigning two outcomes (0 and 1) a probability, we instead assign them a complex number called an *amplitude*. We represent a qubit as a column vector in $\mathbb{C}^2$. For example,

$$\begin{pmatrix} \frac{1}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} \end{pmatrix} \begin{matrix} \leftarrow \text{Amplitude on } 0\text{-outcome} \\ \leftarrow \text{Amplitude on } 1\text{-outcome} \end{matrix}$$

Let's now discuss what it means to "look" at a quantum state, which is called *measurement* in the quantum setting. The measurement axiom of quantum mechanics, called the *Born rule*, says that you see a particular outcome with the squared magnitude of the amplitude. For the example above, this means we'd see the 0-outcome with probability $|1/\sqrt{2}|^2 = 1/2$ and the outcome will be 1 with probability $|i/\sqrt{2}|^2 = 1/2$. Once again, when you observe this outcome the qubit *collapses* to whichever outcome you observed.

From the Born rule, we can derive a condition on the amplitudes of a qubit. Suppose we have a qubit with amplitudes $\alpha, \beta \in \mathbb{C}$. The Born rule states that we see the outcome with probability $|\alpha|^2$ and $|\beta|^2$, respectively. Since there are only two outcomes, these two probabilities must sum up to 1 (i.e., we must see either the O or 1 outcome when we measure). We arrive at the following condition for the amplitudes of a qubit: $|\alpha|^2 + |\beta|^2 = 1$.

Stepping back again, let's give a complete mathematical description of a quantum state. We can generalize to quantum state with d outcomes (called a *qudit* for $d > 2$), which is represented by a length-d complex vector. The key property of this vector is that the squared magnitudes of the amplitudes sum to 1. In other words, the ℓ_2-norm of the vector is 1. The ℓ_2-norm of any $v = (v_1, v_2, \ldots, v_d) \in \mathbb{C}^d$ is defined as

$$\|v\|_2 := \sqrt{\sum_{i=1}^{d} |v_i|^2}.$$

It is an amazing fact that moving from the classical to the quantum setting is in some sense just moving from the ℓ_1 to the ℓ_2 norm.

Unitary matrices

Because the set of valid quantum states must have unit ℓ_2-norm, the set of viable quantum operations must preserve the ℓ_2-norm of the state. However, not all such operations are valid.

An axiom of quantum mechanics dictates that quantum operations must also be *linear*. We will see a slight generalization of this later, but for now, you can think of this linearity as implying that quantum operations are matrices. Applying a quantum operation to a quantum state simply means multiplying the vector of the state with the matrix of the quantum operation.

Matrices preserving the ℓ_2-norm have a beautiful characterization—namely, they are the *unitary* matrices, i.e., matrices $U \in \mathbb{C}^{d\times d}$ such that $UU^\dagger = I$. Here, "$\dagger$" is the conjugate transpose operation and "I" is the identity matrix.

1.2 Multi-qubit quantum computation

In general, we think of large classical computations as a sequence of operations on some bit string. In this way we can break up some large complex operation into a sequence of simpler operations. The number of operations required to build the more complex operation is a proxy for how complex that operation really is. Similarly, in quantum systems, we want to build up larger more complex operations from simpler ones. To do this, we first need to understand what a quantum systems consisting of multiple qubits, so that we can understand what it means to locally apply some quantum operation.

Tensor product of states

Once again, let's start with a discussion of multiple classical random bits, and see how it generalizes to qubits. Let A, B be two random bits. Each bit has some probability of being in the 0 or 1 outcome. Together, the two bits give rise to a probability distribution over pairs of outcomes (i.e., 00, 01, 10, and 11). We can derive the probability of a particular pair of outcomes by multiplying the probabilities of the individual outcome for each bit. For example, let

$$A = \begin{pmatrix} 0.3 \\ 0.7 \end{pmatrix} \begin{matrix} \leftarrow 0 \\ \leftarrow 1 \end{matrix} \qquad , \qquad B = \begin{pmatrix} 0.6 \\ 0.4 \end{pmatrix} \begin{matrix} \leftarrow 0 \\ \leftarrow 1 \end{matrix}$$

Then the product distribution associated to A and B together gives rise to the vector

$$AB = \begin{pmatrix} 0.18 \\ 0.12 \\ 0.42 \\ 0.28 \end{pmatrix} \begin{matrix} \leftarrow 00 \\ \leftarrow 01 \\ \leftarrow 10 \\ \leftarrow 11 \end{matrix}$$

Combining two separate qubits into a single system is exactly the same. Let $v, w \in \mathbb{C}^2$ be vectors representing two qubits. The vector of the joint system is called the *tensor product* $v \otimes w$ of the two vectors v and w. The tensor product operation yields the vector containing all products of amplitudes. The example looks identical to the classical setting:

$$v = \begin{pmatrix} \sqrt{0.3} \\ \sqrt{0.7} \end{pmatrix} \begin{matrix} \leftarrow 0 \\ \leftarrow 1 \end{matrix} \qquad , \qquad w = \begin{pmatrix} \sqrt{0.6} \\ \sqrt{0.4} \end{pmatrix} \begin{matrix} \leftarrow 0 \\ \leftarrow 1 \end{matrix}$$

and

$$v \otimes w = \begin{pmatrix} \sqrt{0.18} \\ \sqrt{0.12} \\ \sqrt{0.42} \\ \sqrt{0.28} \end{pmatrix} \begin{matrix} \leftarrow 00 \\ \leftarrow 01 \\ \leftarrow 10 \\ \leftarrow 11 \end{matrix}$$

Formally, the tensor product operation $\otimes$ is defined over any pair of vectors $v \in \mathbb{C}^a$ and $w \in \mathbb{C}^b$ (not necessarily of the same length) as

$$v \otimes w := \begin{pmatrix} v_1 w \\ \vdots \\ v_a w \end{pmatrix} = \begin{pmatrix} v_1 w_1 \\ \vdots \\ v_1 w_d \\ v_2 w_1 \\ \vdots \\ v_a w_b \end{pmatrix}$$

From this definition, one can derive the following properties of the tensor product, which hold for all complex vectors v, w, z and scalars $\alpha, \beta \in \mathbb{C}$:

Scalar multiplication:	$(\alpha v) \otimes (\beta w) = (\alpha\beta)(v \otimes w)$
Associativity:	$(v \otimes w) \otimes z = v \otimes (w \otimes z)$
Distributivity:	$v \otimes (w + z) = v \otimes w + v \otimes z$

We have that the tensor product of two qubits is represented by a length-4 complex vector, the tensor product of three qubits is represented by a length-8 vector, and so on. One of the key questions we will ask in these notes is: how much of this exponentially is really there? Of course, when it comes to quantum states constructed from tensor products of qubits, the answer is... not much. To describe such a state, we simply need the 2 amplitudes for each individual qubit, a total of $2n$ amplitudes for an n-qubit state, rather than the 2^n amplitudes in the tensor product vector.

Critically, however, not all quantum states over qubits can be described in this way. That is, we can start with tensor product of single-qubit quantum states, apply a sequence of quantum operations, and arrive at a state which cannot be described by any tensor product of single-qubit states. Such states are called *entangled*.

Our first example of an entangled 2-qubit state is the following:

$$\begin{pmatrix} 1/\sqrt{2} \\ 0 \\ 0 \\ 1/\sqrt{2} \end{pmatrix}$$

which is known (amongst other names) as the *Bell state*. Before we prove this state is entangled, let's take a moment to consider what would happen if we measured this state. We would see the 00 outcomes with probability $1/2$ and the 11 outcome with probability $1/2$. In other words, if we made the measurement and we saw that the first qubit was 0, we would immediately know the second qubit was also 0. This description gets even stranger when

we consider the possibility that we could dramatically separate the first and second qubits, putting each on either end of the galaxy (hard to do in practice, of course!). Measuring at one end of the galaxy immediately tells us outcome of the qubit at the other end.[1]

To prove the Bell state is entangled, we argue by contradiction. Suppose otherwise, then we would have

$$\begin{pmatrix} 1/\sqrt{2} \\ 0 \\ 0 \\ 1/\sqrt{2} \end{pmatrix} = \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \otimes \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} = \begin{pmatrix} \alpha_0\beta_0 \\ \alpha_0\beta_1 \\ \alpha_1\beta_0 \\ \alpha_1\beta_1 \end{pmatrix}$$

for some complex amplitudes $\alpha_0, \alpha_1, \beta_0, \beta_1$. Comparing the left and right equations, we get the constraints:

$$\frac{1}{\sqrt{2}} = \alpha_0\beta_0, \;\; 0 = \alpha_0\beta_1, \;\; 0 = \alpha_1\beta_0, \;\; \frac{1}{\sqrt{2}} = \alpha_1\beta_1.$$

One can check this system of equations has no feasible solution, and therefore, the Bell state must entangled.

Tensor product of matrices

The tensor product of matrices is the unique operator which respects the tensor product of the underlying states. That is, for unitaries $U \in \mathbb{C}^a$ and $V \in \mathbb{C}^b$, the tensor product unitary $U \otimes V$ is the unique linear operator such that

$$(U \otimes V)(v \otimes w) = (Uv) \otimes (Vw)$$

for all states $v \in \mathbb{C}^a$ and $w \in \mathbb{C}^b$. This definition lines up with our intuition that if we apply a unitary to a specific qubit, then it should not affect any other qubit.

Formally, one can give a (rather more cumbersome) definition of the tensor product of arbitrary matrices $U \in \mathbb{C}^a$ and $V \in \mathbb{C}^b$ as:

$$U \otimes V = \begin{pmatrix} u_{11}V & u_{12}V & \cdots & u_{1a}V \\ u_{21}V & u_{22}V & \cdots & u_{2a}V \\ \vdots & \vdots & \ddots & \vdots \\ u_{a1}V & u_{a2}V & \cdots & u_{aa}V \end{pmatrix}.$$

Written out somewhat more explicitly when $a = b = 2$, we have

$$U \otimes V = \begin{pmatrix} u_{11}\begin{pmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{pmatrix} & u_{12}\begin{pmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{pmatrix} \\ u_{21}\begin{pmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{pmatrix} & u_{22}\begin{pmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{pmatrix} \end{pmatrix} = \begin{pmatrix} u_{11}v_{11} & u_{11}v_{12} & u_{12}v_{11} & u_{12}v_{12} \\ u_{11}v_{21} & u_{11}v_{22} & u_{12}v_{21} & u_{12}v_{22} \\ u_{21}v_{11} & u_{21}v_{12} & u_{22}v_{11} & u_{22}v_{12} \\ u_{21}v_{21} & u_{21}v_{22} & u_{22}v_{21} & u_{22}v_{22} \end{pmatrix}.$$

[1] A significant amount of ink has been spilled on exactly what is happening at a physical layer when a measurement like this is made. Look up the "quantum measurement problem". Thankfully for one of the most cherished pysical laws, this entanglement phenomenon does *not* allow for faster than light communication.

Partial measurement

With the tensor product, we can now talk about unitary matrices applied to a subset of qubits in our computation. As it turns out, it is also makes sense to measure a subset of qubits. Once again, we can appeal to our classical intuition. Suppose we have the following classical distribution over outcomes:

$$\begin{pmatrix} 0.3 \\ 0.1 \\ 0.3 \\ 0.3 \end{pmatrix} \begin{matrix} \leftarrow 00 \\ \leftarrow 01 \\ \leftarrow 10 \\ \leftarrow 11 \end{matrix}$$

Suppose we look at the second coin, but not the first. The probability the see the 0-outcome for the second coin is

$$\Pr[00\text{-outcome}] + \Pr[10\text{-outcome}] = .3 + .3 = .6$$

since both of those outcomes are consistent with seeing 0 for the second coin. By an identical calculation, we see the 1-outcome for the second coin with 40% probability.

Let's suppose we do see the second coin in the 0-outcome. Now we must calculate the distribution on the first coin conditioned on seeing the second coin in the 0-outcome. For either outcome $b \in \{0, 1\}$, we have

$$\Pr[b \text{ for first coin} \mid 0 \text{ for second coin}] = \frac{\Pr[(b \text{ for first coin}) \wedge (0 \text{ for second coin})]}{\Pr[0 \text{ for second coin}]}.$$

In our example, the probability we see the 0-outcome on the first coin conditioned on having seen 0 for the second outcome is just $.3/.6 = .5$. In practice, its often easiest to do these calculations by simply removing the outcomes that are inconsistent with the partial measurement, and then renormalizing the vector. For our example where we've seen the 0-outcome on the second coin, we have

$$\begin{pmatrix} 0.3 \\ 0.1 \\ 0.3 \\ 0.3 \end{pmatrix} \xrightarrow{\text{Remove inconsistent outcomes}} \begin{pmatrix} 0.3 \\ 0 \\ 0.3 \\ 0 \end{pmatrix} \xrightarrow{\text{Renormalize}} \begin{pmatrix} 0.5 \\ 0 \\ 0.5 \\ 0 \end{pmatrix}.$$

Once again, the quantum setting is identical except everything is done with respect to the ℓ_2-norm rather than the ℓ_1-norm. For completeness, let's look at a similar example with a quantum state:

$$\begin{pmatrix} \sqrt{0.3} \\ \sqrt{0.1} \\ \sqrt{0.3} \\ \sqrt{0.3} \end{pmatrix} \begin{matrix} \leftarrow 00 \\ \leftarrow 01 \\ \leftarrow 10 \\ \leftarrow 11 \end{matrix}$$

The probability we see the 0-outcome for second qubit is $|\sqrt{.3}|^2 + |\sqrt{.3}|^2 = .6$, and the distribution on the first qubit conditioned on this outcome is

$$\begin{pmatrix} \sqrt{0.3} \\ \sqrt{0.1} \\ \sqrt{0.3} \\ \sqrt{0.3} \end{pmatrix} \xrightarrow{\text{Remove inconsistent outcomes}} \begin{pmatrix} \sqrt{0.3} \\ 0 \\ \sqrt{0.3} \\ 0 \end{pmatrix} \xrightarrow{\text{Renormalize}} \begin{pmatrix} \sqrt{0.5} \\ 0 \\ \sqrt{0.5} \\ 0 \end{pmatrix}.$$

This procedure will be easier to describe more formally once we've introduced the notation in the following section.

1.3 Dirac notation and inner products

Let's start this section by introducing a method for writing quantum states, called *Dirac notation*. While this notation may at first seem somewhat unnecessary, it turns out to be quite natural. The most basic notational idea is that we will use a "ket", which looks like $|\cdot\rangle$, to describe a vector that is supposed to be a quantum state (i.e., a unit vector with respect to the ℓ_2-norm). Importantly, we reserve certain vectors special states. In particular, the 0-outcome and 1-outcome states, which we have previously been referring to somewhat awkwardly, are now associated with the following vectors:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \text{and} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

So, for example, we can write an arbitrary single-qubit quantum state $|\psi\rangle$ as

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

for amplitudes $\alpha, \beta \in \mathbb{C}$. To write multi-qubit states in this notation, we employ another useful shorthand for bit strings $x \in \{0,1\}^n$:

$$|x\rangle := |x_1\rangle \otimes |x_2\rangle \otimes \cdots \otimes |x_n\rangle$$

We call such states the *classical basis states*. Now, any n-qubit state $|\psi\rangle$ can be written as linear combination of the classical basis states:

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

where $\alpha_x \in \mathbb{C}$ is some complex amplitude for each $x \in \{0,1\}^n$. For example, we can write the Bell state introduced in the previous section as

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

Inner products

Every quantum state lives in a vector space $\mathbb{C}^d$. We will often use that this vector space is actually a *Hilbert space*, meaning that it is equipped with an inner product: for vectors $v, w \in \mathbb{C}^d$, their *inner product* is defined as

$$v^\dagger w = \sum_{i=1}^{d} \overline{v_i} w_i.$$

In Dirac notation, we write $\langle\psi|$ (pronounced "bra"-ψ) to denote the conjugate transpose of the state $|\psi\rangle$. Therefore, the inner product between two state $|\psi\rangle$ and $|\varphi\rangle$ is written as

$$\langle\psi|\varphi\rangle := \overset{\text{bra}}{\downarrow}\langle\psi| \cdot \overset{\text{ket}}{\downarrow}|\varphi\rangle$$

where the lefthand side shows yet another shorthand. Now we can finally see the reason for the weird names "bra" and "ket". When you put them together to form an inner product, you get the phrase "braket", which looks like "bracket" if you squint.

Why go through all this trouble to create a shorthand for inner products? Perhaps most importantly, the inner product induces a natural distance measure on quantum states. If the inner product of two states is 1, then the states are identical. If the inner product is 0, then the states are perfectly distinguishable.

Outer products

We can also use Dirac notation to denote the outer product between states in the natural way. For states $|\psi\rangle, |\varphi\rangle \in \mathbb{C}^d$, their outer product is

$$|\psi\rangle\langle\varphi| := \begin{pmatrix} \psi_1 \\ \psi_2 \\ \vdots \\ \psi_d \end{pmatrix} \begin{pmatrix} \overline{\varphi_1} & \overline{\varphi_2} & \cdots & \overline{\varphi_d} \end{pmatrix} = \begin{pmatrix} \psi_1\overline{\varphi_1} & \psi_1\overline{\varphi_2} & \cdots & \psi_1\overline{\varphi_d} \\ \psi_2\overline{\varphi_1} & \psi_2\overline{\varphi_2} & \cdots & \psi_2\overline{\varphi_d} \\ \vdots & \vdots & \ddots & \vdots \\ \psi_d\overline{\varphi_1} & \psi_d\overline{\varphi_2} & \cdots & \psi_d\overline{\varphi_d} \end{pmatrix}$$

The outer product is useful for describing quantum operations. For example, an arbitrary n-qubit unitary U can be written as

$$U = \sum_{x,y \in \{0,1\}^n} u_{x,y} |x\rangle\langle y|$$

where $u_{x,y} = \langle x| U |y\rangle \in \mathbb{C}$ is the amplitude the unitary places on the state $|x\rangle$ on input $|y\rangle$. In this case, $|x\rangle\langle y|$ is just matrix which is 1 at entry (x, y) and 0 everywhere else.

Summary – Quantum computation over n qubits

States:	$\|\psi\rangle \in \mathbb{C}^{2^n}$ such that $\sum_{x\in\{0,1\}^n} \|\langle x\|\psi\rangle\|^2 = 1$
Operations:	$U \in \mathbb{C}^{2^n \times 2^n}$ such that $U^\dagger U = U^\dagger U = I$ Applying U to $\|\psi\rangle$ results in the state $U\|\psi\rangle$
Measurement:	State collapses to $\|x\rangle$ with probability $\|\langle x\|\psi\rangle\|^2$

1.4 Mixed states

For many questions in quantum computation, the formalism of states and operations we've previously developed is sufficient. For example, most quantum algorithms start with some classical basis state, apply some unitary operation, and then measure. However, there is actually a more general form of a quantum state that is useful in a variety of contexts, like when you have noise in your quantum computer.

The quantum states $|\psi\rangle$ we have defined previously are called *pure states*. What makes a state "impure", or as it's traditionally called "mixed"? We say that a state is *mixed* when it represents a probability distrubtion of pure states. To see why these two notions are different, it's helpful to look at an example.

On the one hand, let's take the pure state $|+\rangle := \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ which in some sense equal parts $|0\rangle$ and $|1\rangle$. On the other hand, let's take the mixed state which is either $|0\rangle$ or $|1\rangle$ with 50% probability. These states may superficially seem to be the same (after all, they have the same probability over outcomes when measured), but are actually quite different. To see this, let's examine what happens when we apply the following unitary H, which is called the *Hadamard gate*:

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

Applying H to our pure state $|+\rangle$, we get

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

In other words, if we were to measure our pure state *after* the application of the unitary operation H, then we are guaranteed to see the outcome $|0\rangle$. This will not be true in our mixed state picture. Let's do the calculation. Applying H to the mixed state, we get

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

and

$$H|1\rangle = \frac{1}{2} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} =: |-\rangle,$$

each of which happens with 50% probability. What is the probability we measure $|0\rangle$ now? Given the calculation above of what the Hadamard transformation does to each of our starting states, we have

$$\begin{aligned} \Pr[\text{measure } |0\rangle] =& \Pr[\text{Original state was } |0\rangle] \cdot \Pr[\text{measure } |0\rangle \text{ on state } |+\rangle] \\ &+ \Pr[\text{Original state was } |1\rangle] \cdot \Pr[\text{measure } |0\rangle \text{ on state } |-\rangle] \\ =& \frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2} \end{aligned}$$

We can now see that when our state was a statistical mixture of $|0\rangle$ and $|1\rangle$, the Hadamard transformation didn't change our measurement probabilities at all. In fact, this is a general phenomenon. One can show that no matter what unitary transformation you apply to this mixed state, you will always get $|0\rangle$ and $|1\rangle$ with 50% probability. This will be easy to show using the formalism we now introduce.

Density matrices

General quantum systems are fully described by statistical mixtures of quantum states—that is, an ensemble of pure states $\{|\psi_i\rangle\}_i$ each of which is prepared with probability $p_i \in [0, 1]$. The *density matrix* corresponding to this ensemble is

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i| \in \mathbb{C}^{2^n \times 2^n}$$

where $\sum_i p_i = 1$. One can show that if you have a density matrix ρ and apply a unitary U, that the new density matrix is given by $U\rho U^\dagger$. Furthermore, measurement results in outcome $|x\rangle$ with probability $\langle x| \rho |x\rangle$, whereupon ρ collapses to the state $|x\rangle\langle x|$.

Let's revisit our example of an even statistical mixture of the states $|0\rangle$ and $|1\rangle$. The corresponding density matrix is

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

As it turns out this stated is called the *maximally mixed state* since it represents that we essentially have no knowledge of what the underlying state is. To see this, imagine applying any unitary U to this state. We would get

$$U\left(\frac{I}{2}\right)U^\dagger = \frac{UU^\dagger}{2} = \frac{I}{2},$$

as the new state, which is the same state we started with. In other words, no unitary operation changes how the state looks. This proves the claim we made earlier that any unitary followed by measurement would result in outcomes $|0\rangle$ and $|1\rangle$ with equal probability.

What matrices correspond to ensembles of pure states? As it turns out, there is a very nice characterization: ρ is a valid density matrix if and only if ρ is a trace-1 positive semidefinite matrix. Trace-1 implies that $\mathrm{Tr}(\rho) = 1$. Positive semidefinite implies that $\langle\psi| \rho |\psi\rangle \geq 0$ for all pure states $|\psi\rangle$.

The forward direction of this claim can be shown by reasoning directly about the types of matrices that an ensemble of states can give rise to. The reverse direction can be shown by taking the spectral decomposition of ρ, which is valid since we have assumed that ρ is positive semidefinite. The eigenvectors of this decomposition will be the pure states in the decomposition, and the eigenvalues will be the associated probabilities.

Quantum channels

As one might have now guessed, unitary transformations are also not the most general transformation on quantum states. Quantum transformations that work on the level of density matrices are called *quantum channels*. That said, it is not true that every channel which preserves density matrices corresponds to a valid quantum operation. Most importantly, as required by the axioms of quantum mechanics, the channel must be linear. Furthermore, for technical reasons having to do with applying the channel to a restricted set of qubits, we must also require that the quantum channel still maps density matrices to density matrices when it is tensored with the identity map. Maps satisfying all the above conditions are called *completely positive trace preserving (CPTP)*.

Measurement

While there is a more general form of quantum measurements, it turns out that these more general measurements can be simulated by the measurements that we have already introduced. So, for simplicity, we will always assume that we measure our qubits the usual way.

Summary – Quantum computation with n-qubit mixed states

States:	$\rho \in \mathbb{C}^{2^n \times 2^n}$ such that $\mathrm{Tr}(\rho) = 1$ and ρ is positive semidefinite
Operations:	Completely positive trace-preserving maps Φ If Φ is a unitary channel, then $\Phi(\rho) = U\rho U^\dagger$ for unitary $U \in \mathbb{C}^{2^n \times 2^n}$
Measurement:	State collapses to $\lvert x\rangle\langle x\rvert$ with probability $\langle x\rvert \rho \lvert x\rangle$

Partial Trace

One of the most important reasons to introduce the density matrix formalism is to be able to talk about parts of a quantum state in isolation. That is, even if we have an n-qubit pure state, it is not necessarily the case that the state restricted to, say, the first $n/2$ qubits is a pure state.

We now introduce a way to "trace out" part of a density matrix of a large system to describe the state on the leftover qubits. To start, let's imagine we start with a composite system $\mathcal{H}_A \otimes \mathcal{H}_B$. For simplicitly, you can at first just assume that $\mathcal{H}_A$ and $\mathcal{H}_B$ are the Hilbert spaces for two different qubits. Formally, the partial trace Tr_B is the unique linear map satisfying

$$\mathrm{Tr}_B(|a_i\rangle\langle a_j| \otimes |b_i\rangle\langle b_j|) = |a_i\rangle\langle a_j| \mathrm{Tr}(|b_i\rangle\langle b_j|),$$

where $a_i, a_j \in \mathcal{H}_A$ and $b_i, b_j \in \mathcal{H}_B$ are basis elements for the two subsystems.

So, if we have some state ρ_{AB} that lives in the Hilbert space $\mathcal{H}_A \otimes \mathcal{H}_B$, then the density matrix for the subsystem A after ignoring the subsystem B is given by

$$\rho_A = \mathrm{Tr}_B(\rho_{AB}).$$

If we apply the partial trace operator to a product state we get, unsurprisingly,

$$\mathrm{Tr}_B(\rho_A \otimes \rho_B) = \rho_A.$$

What happens when we take the partial trace of the Bell state? The density matrix is given by

$$\rho_{\text{Bell}} := \left(\frac{|00\rangle + |11\rangle}{\sqrt{2}}\right)\left(\frac{\langle 00| + \langle 11|}{\sqrt{2}}\right) = \frac{|00\rangle\langle 00| + |00\rangle\langle 11| + |11\rangle\langle 00| + |11\rangle\langle 11|}{2}$$

so tracing out the second qubit, we get (by linearity of the partial trace)

$$\begin{aligned}
\mathrm{Tr}_2(\rho_{\text{Bell}}) &= \frac{1}{2}\left(\mathrm{Tr}_2(|00\rangle\langle 00|) + \mathrm{Tr}_2(|00\rangle\langle 11|) + \mathrm{Tr}_2(|11\rangle\langle 00|) + \mathrm{Tr}_2(|11\rangle\langle 11|)\right) \\
&= \frac{1}{2}\left(|0\rangle\langle 0| \mathrm{Tr}(|0\rangle\langle 0|) + |0\rangle\langle 1| \mathrm{Tr}(|0\rangle\langle 1|) + |1\rangle\langle 0| \mathrm{Tr}(|1\rangle\langle 0|) + |1\rangle\langle 1| \mathrm{Tr}(|1\rangle\langle 1|)\right) \\
&= \frac{1}{2}\left(|0\rangle\langle 0| \cdot 1 + |0\rangle\langle 1| \cdot 0 + |1\rangle\langle 0| \cdot 0 + |1\rangle\langle 1| \cdot 1\right) \\
&= \frac{|0\rangle\langle 0| + |1\rangle\langle 1|}{2}.
\end{aligned}$$

That is, if we take the Bell state and trace out a qubit, we are left with the maximally mixed state. This may give you some sense of the fragility of quantum computations. If you take an entangled state and lose a single qubit, it may become completely useless.

Reconciling the pure and mixed states

Often it will be easier to reason about pure states rather than mixed ones. As we've seen before, this is in some sense fundamentally impossible—there are mixed states which behave completely differently from pure ones. That said, there is also some sense in which there is an equivalence between the two settings. Namely, for every n-qubit mixed state ρ, there is a $(2n)$-qubit pure state such that tracing out the last n qubits of $|\psi\rangle$ leaves the state ρ. This process is called *purification*.

We will give an explicit purification procedure. First, let ρ be an arbitrary n-qubit mixed state:

$$\rho = \sum_{x\in\{0,1\}^n} p_x \, |\psi_x\rangle\langle\psi_x|$$

The following state will be a purification of ρ:

$$|\psi\rangle \coloneqq \sum_{x\in\{0,1\}^n} \sqrt{p_x} \, |\psi_x\rangle \otimes |x\rangle$$

Let B be the system consisting of the last n qubits. Tracing out B, we get the density matrix:

$$\begin{aligned}
\mathrm{Tr}_B\left(|\psi\rangle\langle\psi|\right) &= \mathrm{Tr}_B\left(\textstyle\sum_{x,y} \sqrt{p_x p_y}\, |\psi_x\rangle\langle\psi_y| \otimes |x\rangle\langle y|\right) \\
&= \textstyle\sum_{x,y} \sqrt{p_x p_y}\, \mathrm{Tr}_B(|\psi_x\rangle\langle\psi_y| \otimes |x\rangle\langle y|) && \text{(Linearity of partial trace)} \\
&= \textstyle\sum_{x,y} \sqrt{p_x p_y}\, |\psi_x\rangle\langle\psi_j| \,\mathrm{Tr}\left(|x\rangle\langle y|\right) && \text{(Definition of partial trace)} \\
&= \textstyle\sum_x p_x \, |\psi_x\rangle\langle\psi_x| && \text{(Trace is 1 iff } x = y\text{)}
\end{aligned}$$

which is precisely the mixed state ρ that we wanted to embed into $|\psi\rangle$.

Are purifications unique? Unfortunately, not. To see this, notice that we can generalize our purification procedure above my multiplying the second register by any n-qubit unitary U:

$$|\psi\rangle \coloneqq \sum_{x\in\{0,1\}^n} \sqrt{p_x} \, |\psi_x\rangle \otimes (U\,|x\rangle).$$

Intuitively, it makes sense that changing the basis of the second register shouldn't affect partial trace since we never used anything special about the classical basis states. Formally, you can check that the computation is agnostic to the choice of unitary U because of the following equalities:

$$\mathrm{Tr}(U\,|x\rangle\langle y|\,U^\dagger) = \mathrm{Tr}(U^\dagger U\,|x\rangle\langle y|) = \mathrm{Tr}(|x\rangle\langle y|)$$

where the first equality uses the cyclic property of the trace and the second using the fact that U is unitary.

1.5 Noteworthy quantum phenomena

Let's start to use the quantum formalism to take note of some interesting phenomena. We start with a classic result which implies that quantum information cannot be copied.

Theorem 1 (No-Cloning Theorem)**.** *There is no $(2n)$-qubit unitary U and n-qubit state $|\varphi\rangle$ such that*

$$U(|\psi\rangle \otimes |\varphi\rangle) = |\psi\rangle \otimes |\psi\rangle$$

for all pure states $|\psi\rangle$.

Proof. We argue by constradiction. Suppose such at U and $|\varphi\rangle$ existed, and let $|\psi_1\rangle, |\psi_2\rangle$ be two states we want to copy. In other words, we have

$$U(|\psi_i\rangle \otimes |\varphi\rangle) = |\psi_i\rangle \otimes |\psi_i\rangle$$

for $i \in \{1, 2\}$. Let's now take the inner product of the two states $U(|\psi_1\rangle \otimes |\varphi\rangle)$ and $U(|\psi_2\rangle \otimes |\varphi\rangle)$

$$(|\psi_1\rangle \otimes |\varphi\rangle)U^\dagger U(|\psi_2\rangle \otimes |\varphi\rangle) = \langle\psi_1|\psi_2\rangle \langle\varphi|\varphi\rangle = \langle\psi_1|\psi_2\rangle$$

and compare it to the inner product of $|\psi_1\rangle \otimes |\psi_1\rangle$ and $|\psi_2\rangle \otimes |\psi_2\rangle$:

$$(\langle\psi_1| \otimes \langle\psi_1|)(|\psi_2\rangle \otimes |\psi_2\rangle) = \langle\psi_1|\psi_2\rangle^2.$$

Cloning implies that these two expressions are equal:

$$\langle\psi_1|\psi_2\rangle = \langle\psi_1|\psi_2\rangle^2.$$

However, for any states such that $\langle\psi_1|\psi_2\rangle \notin \{0, 1\}$, the above equation will not hold. That is, cloning breaks for any distinct pair of non-orthogonal states! □

Chapter 2

Computation with Quantum Circuits

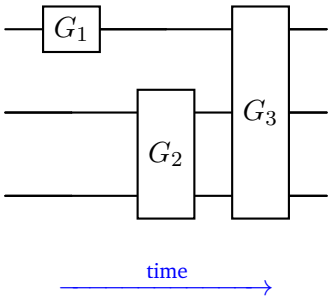
2.1 The quantum circuit model

How do we describe a quantum algorithm? One might think that something like a generalization of the classical Turing machine may be a particularly apt choice, given the centrality of that model to the story of classical theory of computation. While it is possible to define a quantum Turing machine, it turns out to be rather cumbersome to work with.

Instead, we will use a model of computation that more-or-less is the straightforward realization of applying a sequence of unitaries—the *quantum circuit*.

Introduction to quantum circuits

A n-qubit quantum circuit is a collection of unitary operations $G_1, \ldots, G_m$, called *gates*, applied in sequence to a subset of n *wires*. The composition of the gates in the circuit generates a $2^n \times 2^n$ unitary operation. We assume that each gate is in tensor product with the identity operation on each wire that it does not touch. Let's look at a simple example:



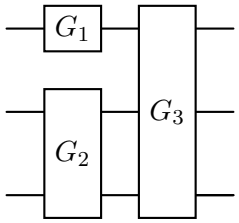
The above diagram is a circut on 3 qubits with 3 gates: the single-qubit gate G_1 is applied first; the 2-qubit gate G_2 is applied next; and finally G_3 is applied as a 3-qubit gate. The unitary matrix representing this circuit is

$$G_3 \left(I \otimes G_2\right) \left(G_1 \otimes I \otimes I\right).$$

Beware: matrix multiplication happens the reverse order of the circuit, which is why G_1 appears last the composition of unitaries. Since G_1 and G_2 act on different wires, we get that

$$\left(I \otimes G_2\right) \left(G_1 \otimes I \otimes I\right) = G_1 \otimes G_2.$$

Therefore, in the diagram, we can put G_1 and G_2 on the same *layer*.



That is, a layer of the circuits consists of a set of gates that can be applied simultaneously since they act on different qubits. The *depth* of a circuit is the number of layers of gates it has. Therefore, the example circuit above has depth 2.

Examples with common gates

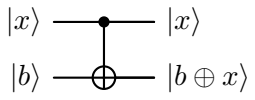
Let's take a look at some of the most common gates used in quantum circuits and the special notation that we use to denote them.

Classical reversible gates

One of the most common two-qubit gates is the *controlled-NOT* or *CNOT* gate. Recall that by linearity, it suffices to define the action of any gate on the computational basis. CNOT has the following action:

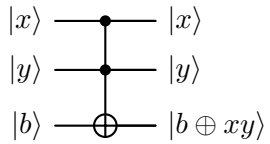
$$|00\rangle \mapsto |00\rangle\,, \quad |01\rangle \mapsto |01\rangle\,, \quad |10\rangle \mapsto |11\rangle\,, \quad |11\rangle \mapsto |10\rangle\,.$$

Notice that CNOT maps any computational basis state to another computational basis state. That is, the CNOT gate is "classical" in the sense that it cannot be used to create superposition of inputs. A CNOT gate in a circuit is depicted as a $\bullet$ symbol (the *control*) connected to a $\oplus$ symbol (the *target*):



Here, we've shown how the CNOT gate acts on general computational basis states, where $x, b \in \{0, 1\}$ are arbitrary bits and $b \oplus x$ denotes their XOR (i.e., addition modulo 2).

Another related gate is the version of the CNOT gate with an extra control, that is, the *controlled-controlled-NOT* gate, most commonly referred to as the *Toffoli* gate. As a circuit, it looks like



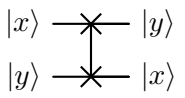
where $x, y, b \in \{0, 1\}$ are arbitrary bits (xy is the product of x and y). Notice that the third bit is flipped exactly when both controls are 1.

The Toffoli gate is in some sense more powerful than the CNOT gate since it can be used to generate the CNOT gate. Notice that if we set the second input qubit above to $|1\rangle$ (i.e., $y = 1$), then the remaining effect on the remaining two qubits is exactly the CNOT gate. We will see later however, that the reverse is not true—we cannot just use the CNOT gate to generate a Toffoli gate.

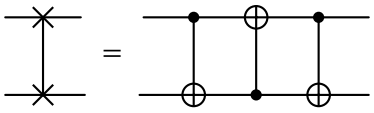
Finally, let's discuss the SWAP gate, another important "classical reversible" gate on 2 qubits. Aplty named, the SWAP gate swaps qubits, i.e., for all $x, y \in \{0, 1\}$ it maps:

$$|xy\rangle \mapsto |yx\rangle .$$

In a circuit diagram, it is depicted as



One can check the following nice identity:



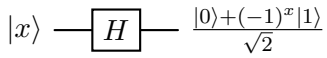
In other words, we can replace every SWAP gate in a circuit with 3 CNOT gates. This is a common theme we will continue to see—we can take some gates as the fundamental ones that will generate the rest.

Change of basis operations

Evidently, we need a gate that can create a superposition of inputs from a classical basis state. The *Hadamard gate* is the canonical choice for such an operation. It has the action

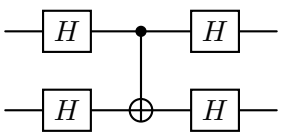
$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} := |+\rangle \qquad\qquad H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} := |-\rangle$$

on the computational basis. Notice that Hadamard gate has given rise to a new basis, the $\{|+\rangle, |-\rangle\}$ basis. In fact, Hadamard switches back and forth between the computational basis and this new basis. That is, the Hadamard gate is its own inverse: $H^2 = I$. As a circuit, it is shown as



for any $x \in \{0, 1\}$.

As another example, let's consider a circuit built from Hadamard and CNOT gates:



One way of understanding this circuit would be to just explicitly compute the unitary matrix $(H \otimes H)\mathrm{CNOT}(H \otimes H)$, but it is often more helpful to instead look at how the system evolves over a basis. Let's see how it acts on the computational basis, considering one gate at a time:

$$|00\rangle \xrightarrow{H\otimes H} |+\rangle \otimes |+\rangle = \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) = \frac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{2}$$

That is, after applying $H \otimes H$, we have the uniform superposition over 2-qubit computational basis states. We know that the CNOT gate just permutes the elements of the computational basis, or, in other words, it must do nothing to do the above state:

$$\frac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{2} \xrightarrow{\mathrm{CNOT}} \frac{|00\rangle + |01\rangle + |10\rangle + |11\rangle}{2}.$$

Of course, if we've done nothing to the state, then it must also factorize as

$$|+\rangle \otimes |+\rangle = (H \otimes H)\,|00\rangle\,.$$

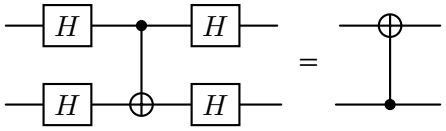
Therefore, the final layer of Hadamard gates returns the state to $|00\rangle$. That is, after all that computation, we see that the circuit acts as the identity on the $|00\rangle$. For completeness, let's see one more case (the input $|01\rangle$) in its entirety:

$$\begin{aligned} |01\rangle &\xrightarrow{H\otimes H} \left(\frac{|0\rangle + |1\rangle}{\sqrt{2}}\right) \otimes \left(\frac{|0\rangle - |1\rangle}{\sqrt{2}}\right) = \frac{|00\rangle - |01\rangle + |10\rangle - |11\rangle}{2} \\ &\xrightarrow{\mathrm{CNOT}} \frac{|00\rangle - |01\rangle - |10\rangle + |11\rangle}{2} = |-\rangle \otimes |-\rangle \\ &\xrightarrow{H\otimes H} |11\rangle \end{aligned}$$

If we were to continue with the entire computational basis, we would see

$$|00\rangle \mapsto |00\rangle\,, \quad |01\rangle \mapsto |11\rangle\,, \quad |10\rangle \mapsto |10\rangle\,, \quad |11\rangle \mapsto |01\rangle\,.$$

We've seen this gate before. It's just the CNOT gate with the control on the second qubit instead of the first! That is, we've derived the following circuit identity:



Notice that up until this point, every gate that we've introduced is *real*—the all elements of the unitary matrix representing the gate are real numbers. Let's now introduce some gates that have complex entries.

Phase gates

The gate most commonly referred to as the "phase gate" is the single-qubit diagonal gate S that simply multiplies the $|1\rangle$ state by a phase of i:

$$S\,|0\rangle = |0\rangle \qquad\qquad S\,|1\rangle = i\,|1\rangle$$

Another type of phase gate, most commonly called a T-gate, is the square root of this operation:

$$T|0\rangle = |0\rangle \qquad T|1\rangle = e^{i\frac{\pi}{4}}|1\rangle$$

Unsurprisingly, there are many other common gates that we have yet to define. Thankfully, the gates we have already allow us to do essentially everything we want.

Universality, approximations, and circuit size

A *gate set* is the collections of gates that one can use in the construction of a circuit. Typically, when a gate is included in a gate set, then you're allowed to apply that gate as many times you like on whichever subset of qubits that you like.

Universality captures the notion that a particular gate set can be used to construct any possible quantum operation. There are several different kinds of universality you might want:

- *Exact Universality:* For any n-qubit unitary, there is a circuit that exactly compute the unitary.
- *Approximate Universality:* For any n-qubit unitary, there is a circuit that approximately computes the unitary. One common measure of closeness is the operator norm.
- *Computational Universality:* For any n-qubit unitary, the probability distribution resulting from measuring the first qubit can be approximated by measuring the first qubit of the circuit. For example, it turns out that real quantum gates (without complex entries) are sufficient for computational universality, whereas they clearly fail on the other two notions of universality.

Given that we have a universal gate set, how many gates do we actually need to construct an arbitrary unitary? Let's look at the exact case, where we can get an estimate based on the number of parameters it takes to specify arbitrary unitary matrix. The first claim is that an arbitrary complex $d \times d$ unitary matrix U is specified by d^2 real parameters.

To see this, first note there are d^2 entries in the matrix, each with a complex part and a real part, that is, $2d^2$ real parameters total. However, the unitary constraint $UU^\dagger = I$ imposes d^2 algebraically independent real conditions (d for the fact that the norm of each column should be 1, and $d(d-1)$ conditions for the fact that the complex inner product of each row should be 0).

If our gate set consists of gates that only act on a constant number of qubits (which is often the convention), then each such gate only contributes constantly many real parameters to the construction of the unitary. Therefore, we must have $\Omega(4^n)$ gates to construct an arbitrary n-qubit unitary exactly.

A generalization of this result shows that this lower bound is essentially tight for approximation computation as well—$\Omega(4^n \log(1/\epsilon))$ gates are required to approximately compute any unitary to within ϵ-accuracy with respect to the operator norm [DN06]. That is, a unitary U is ϵ-close to unitary V if

$$\|U - V\|_{\text{op}} = \sup_{\psi} \|(U - V)|\psi\rangle\|_2 \leq \epsilon.$$

Thankfully, there is a matching (up to polylog factors) circuit building algorithm as well.

Theorem 2 (Solovay-Kitaev [Kit97])**.** *Given an approximately universal gate set, there is a circuit to appoximate any unitary to ϵ-accuracy with $\mathcal{O}(4^n \mathsf{polylog}(\frac{1}{\epsilon}))$ gates.*

The version of the Solovay-Kitaev theorem stated above is actually the result of Bouland and Giurgica-Tiron [BGT21], who show how to work with general gate sets. Unfortunately, their result suffers in the exponent of the log factor. The original and most-efficient Solovay-Kitaev theorems require that if a gate is in the gate set, then its inverse is also in the gate set. The current best result in this setting is by Kuperberg, who shows a bound of $O(4^n \log^{1.441}(1/\epsilon))$ gates [Kup23].

2.2 The complexity class BQP

This section relies on a background in classical complexity theory. For a short review of some of the fundamental classical complexity classes, see Appendix A.

To properly define the quantum complexity class BQP, we need to first discuss how a quantum circuit is encoded. Let us suppose that the circuit is constructed from some reasonable universal gate set (i.e., all the amplitudes used in the gates are efficiently computable). We will use the notation $\langle Q \rangle$ to denote the encoding of a circuit Q as a bit string. We now discuss the requirement for a proper encoding:

1. The encoding is unique: mapping from a circuit to its encoding must be *injective*.
2. The encoding is not too big: if Q has m gates, then $\langle Q \rangle$ has at most $\mathsf{poly}(m)$ bits.
3. The encoding is not too small: if Q has m gates, then $\langle Q \rangle$ has at least m bits.

Equipped with an encoding, we can now talk about Turing machines whose output is (an encoding of) a quantum circuit. A circuit family $\{Q_n\}_{n=1}^{\infty}$ is *poly-time uniform* if there exists a poly-time Turing machine such that on input 1^n outputs $\langle Q_n \rangle$. We are now ready to define the complexity class capturing efficient quantum computation:

Bounded-error Quantum Polynomial Time (BQP):
Languages L such that there exists poly-time uniform class of quantum circuits $\{Q_n\}_{n=1}^{\infty}$ and a polynomial q such that for all $x \in \{0,1\}^n$:

- If $x \in L$, probability of measuring $|1\rangle$ on the first qubit of $Q_n |x\rangle \otimes |0^{q(n)}\rangle$ is at least $\frac{2}{3}$.
- If $x \notin L$, probability of measuring $|1\rangle$ on the first qubit of $Q_n |x\rangle \otimes |0^{q(n)}\rangle$ is at most $\frac{1}{3}$.

Bibliography

[BGT21] Adam Bouland and Tudor Giurgica-Tiron. Efficient universal quantum compilation: An inverse-free Solovay-Kitaev algorithm. *arXiv preprint arXiv:2112.02040*, 2021.

[DN06] Christopher M Dawson and Michael A Nielsen. The Solovay-Kitaev algorithm. *Quantum Information & Computation*, 6(1):81–95, 2006.

[Kit97] A. Y. Kitaev. Quantum computations: algorithms and error correction. *Russ. Math. Surv.*, 52(6):1191–1249, 1997.

[Kup23] Greg Kuperberg. Breaking the cubic barrier in the Solovay-Kitaev algorithm. *arXiv preprint arXiv:2306.13158*, 2023.

Appendix A

Classical complexity

In this appendix, we review some of the important concepts from classical complexity theory.

A.1 Complexity classes for decision problems

For this section, let's suppose all computation is done over the binary alphabet $\{0,1\}^*$. A *language* $L \subseteq \{0,1\}^*$ is simply a set of strings. In this section, we will focus on decision problems, where the goal is the compute membership in language or promise language. A *complexity class* is a collection of languages recognized by a particular model of computation.

The complexity classes below are defined in terms of classical Turing machines. The more resources we give the Turing machine (e.g., time, space, randomness, or nondeterminism), the more languages that model of Turing machine can recognize.

Polynomial Time (P):
Languages L such that there exists a deterministic poly-time Turing machine M such that M accepts x iff $x \in L$.

Non-deterministic Polynomial Time (NP):
Language L such that there exists deterministic poly-time Turing machine M and polynomial q such that for all $x \in \{0,1\}^n$

- If $x \in L$, $\exists y \in \{0,1\}^{q(n)}$ such that $M(x,y)$ accepts
- If $x \notin L$, $\forall y \in \{0,1\}^{q(n)}$, $M(x,y)$ rejects.

NP is a generalization of P (just forget about the witness string y), so $\mathsf{P} \subseteq \mathsf{NP}$. It is widely conjectured that $\mathsf{P} \neq \mathsf{NP}$, but we do not have a proof!

Bounded-error Probabalistic Polynomial Time (BPP):
Languages L such that there exists a deterministic poly-time Turing machine M and polynomial q such that for all $x \in \{0,1\}^n$

- If $x \in L$, then $M(x,y)$ accepts for at least $2/3$ of the strings $y \in \{0,1\}^{q(n)}$.
- If $x \notin L$, then $M(x,y)$ accepts for at most $1/3$ of the strings $y \in \{0,1\}^{q(n)}$.

That is, y is a random string given to the Turing machine. Once again, it is clear that $\mathsf{P} \subseteq \mathsf{BPP}$ since we can just forget about the extra random bits. However, we do not know if $\mathsf{BPP} \subseteq \mathsf{NP}$ or if $\mathsf{NP} \subseteq \mathsf{BPP}$, though it is widely conjectured that $\mathsf{P} = \mathsf{BPP}$.

Probabilistic Polynomial Time (PP):
Languages L such that there exists a deterministic poly-time Turing machine M and polynomial q such that for all $x \in \{0,1\}^n$

- If $x \in L$, then $M(x,y)$ accepts for more than $1/2$ of strings $y \in \{0,1\}^{q(n)}$.
- If $x \notin L$, then $M(x,y)$ accepts at most $1/2$ of strings $y \in \{0,1\}^{q(n)}$.

Notice that the definition PP to identical to that of BPP except with a smaller gap between acceptance and rejection probabilities. Therefore, we have that $\mathsf{BPP} \subseteq \mathsf{PP}$. In fact, PP is powerful enough even to contain NP. To see this, take any NP machine M, and alter it in the following way. If $M(x,y)$ accepts, then accept. If $M(x,y)$ rejects, then flip an unbiased coin (to be completely rigorous, one would need to extend the length of the random string y)—if heads, accept, and if tails, reject. Notice that if there are no accepting y for the original machine, then the new machine accepts with exactly 50% probability. However, if there is any accepting y, then the new machine accepts with greater than 50%, and so the inclusion $\mathsf{NP} \subseteq \mathsf{PP}$ follows.

Polynomial Space (PSPACE):
Languages L such that there exists a deterministic Turing machine M that uses at most polynomial space and M accepts x iff $x \in L$.

We have that $\mathsf{PP} \subseteq \mathsf{PSPACE}$ since a PSPACE machine can simply count all the $y \in \{0,1\}^{q(n)}$ that make a poly-time Turing machine accept. There are exponentially many such y, but this is not an issue since we can erase the previous computation as we are enumerating over all the y.

Exponential Time (EXP):
Languages L such that there exists a deterministic Turing machine M and a polynomial q such M halts in $2^{q(n)}$ time and M accepts x iff $x \in L$.

We have that $\mathsf{PSPACE} \subseteq \mathsf{EXP}$ because a Turing machine that uses polynomial space can only have exponentially many configurations. And, if you were to reach the same configuration twice, then you will be in an infinite loop, so you might as well halt.

Figure A.1 shows a summary of how the complexity classes introduced above relate to each other.

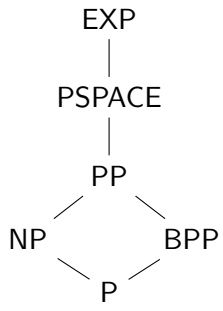


Figure A.1: Inclusion diagram of classical complexity classes. A is below B if $A \subseteq B$.