HSSP Week 4 Class Notes: More Quantum Gates, Deutsch's Algorithm

Agustin Valdes Martinez

March 2024

1 Recap

Last class, we took our general definition for a qubit, the classical NOT gate, and combined them to figure out what the 'quantum' NOT gate must be.

1.1 Qubits

The most general qubit can be written as

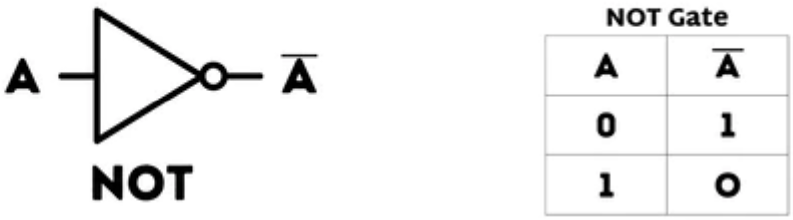
$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1}$$

where $|\alpha|^2$ is the probability that the qubit is in the $|0\rangle$ state and $|\beta|^2$ is the probability that the qubit is in the $|1\rangle$ state. This means that α and β themselves are not probabilities; rather, they are called *probability amplitudes*.

What kind of mathematical object is a qubit? Well recall that we've defined $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Plugging these into Equation 1 we see that our qubit is a vector with **two entries**, one for each probability amplitude: $|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$.

1.2 Quantum NOT Gate

You can imagine it'd be useful for computation to have an operation on our qubit that takes $|0\rangle \to |1\rangle$ and $|1\rangle \to |0\rangle$. This is very similar to the behavior you'd expect out of the *classical* NOT gate (Figure 1).



NOT Gate

A	$\overline{A}$
0	1
1	0

Figure 1: The classical NOT gate is just the bit flip operation.

After some trial and error, we found last class that the 'quantum' NOT gate must be the following 2x2 matrix:

$$X = \sigma_1 = \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{2}$$

This quantum logic gate goes by many names (σ_x, σ_1, X); regardless, it's one of the famous *Pauli matrices*, which come up all the time in quantum computation and the physics of spin- $\frac{1}{2}$ particles.

Let's check to see that this matrix works the way we'd expect:

$$X|0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |1\rangle \checkmark \tag{3}$$

$$X|1\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle \checkmark \tag{4}$$

Good! As one of you pointed out last class, the X gate doesn't just act on the $|0\rangle$ and $|1\rangle$ states. **It can also act on *superpositions* (sums) of them**. That means that it works for even the most general qubit we can construct. To confirm this is true, let's see what it does to the qubit in Equation 1:

$$X|\psi\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix} \tag{5}$$

It flipped the entries! Indeed, we've found ourselves the 'quantum' bit flip!

2 More Quantum Gates

2.1 The Paulis

We've already met the first Pauli matrix: X; but actually, we have 3 more to meet. Instead of deriving them all like we did with X, I'll just list them out and we'll see how they act on our most general qubit state (Equation 1):

2.1.1 Identity

This is the 'do nothing' operation:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{6}$$

$$I|\psi\rangle = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix} \tag{7}$$

In Equation 7, we apply the identity gate to our qubit, and get the same qubit back!

2.1.2 'Quantum' Bit Flip

We've met this guy already! Action from the X gate will exchange the entries in the qubit:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{8}$$

$$X|\psi\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix} \tag{9}$$

2.1.3 'Quantum' Phase Flip

Unlike regular bits, qubits can have *phases*, which manifests here with a negative sign on the second entry

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{10}$$

$$Z|\psi\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ -\beta \end{pmatrix} \tag{11}$$

2.1.4 'Quantum' Phase *and* Bit Flips

You can think of the Y gate as having both a bit flipping and phase flipping effect.

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{12}$$

$$Y|\psi\rangle = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} -i\beta \\ i\alpha \end{pmatrix} = i \begin{pmatrix} -\beta \\ \alpha \end{pmatrix} \tag{13}$$

Not only does it exchange the entries in the qubit (just like the X gate), but it also introduces a phase via the negative sign (similar to the Z gate).

2.2 Hadamard

The Hadamard gate is another essential quantum gate to know. It's technically a special case of the *quantum Fourier transform*, but we won't delve into that here. In matrix form, it's written as

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \tag{14}$$

3 Our First Quantum Algorithm

So we have all of these quantum logic gates... what can we do with them? You might have this vague notion that if we string them together in a special sequence, we might be able to do something interesting to our qubit. This is exactly the idea behind quantum algorithms.

In case you don't know, an algorithm is a set of instructions that leads to some result. We will now discuss one of the earliest quantum algorithms developed which showed some advantage over classical algorithms: **Deutsch's algorithm**.

3.1 The Problem

Every algorithm intends to solve some problem, which we need to fully understand before seeing how, in this case, Deutsch's algorithm solves it. Suppose I hand you a function $f(x)$, **which takes in as input either 0 or 1**, and I tell you that it's either *constant* or *balanced*.

- Constant means all inputs map to the same output: $f(0) = f(1)$. This means either $f(0) = f(1) = 0$ OR $f(0) = f(1) = 1$.
- Balanced means one input maps to 0, and the other maps to 1. So $f(0) \neq f(1)$ and either $f(0) = 0, f(1) = 1$ OR $f(0) = 1, f(1) = 0$.

The problem is to figure out whether f(x) is balanced or constant by evaluating f(x) as few times as possible!

3.2 Oracles and Performance of Algorithms

To say one algorithm is better than another, we need a metric to compare them. A common choice for this problem is **the number of queries to an oracle**. What does this mean?

Imagine we have a black box that, when we query it, we get a result back. This is how we will evaluate $f(x)$: we assume there is a black box 'oracle' that, if I give it x, it returns $f(x)$.

We'll see that in the classical solution to this problem, it takes **two** queries to the oracles to know whether $f(x)$ is balanced or not. In the quantum case, it'll take only **one**! We will use the following oracle:

$$O_f|x\rangle = (-1)^{f(x)}|x\rangle \tag{15}$$

It might seem weird to not write it out as a matrix, as we have with all of our other operations/gates, but usually, oracles are written out this way to show how they act on some state $|x\rangle$.

We'll get practice using this oracle in a bit.

3.3 Classic Solution

How might I solve this with a classical computer? Well, I will need to evaluate **both** $f(0)$ and $f(1)$ to determine whether $f(x)$ is constant or balanced. That's two queries to the oracle!

3.4 Quantum Solution - Deutsch's Algorithm

Again, a quantum algorithm is just a sequence of quantum logic gates we need to apply to an initial state to reach our solution: in this case, figure out whether $f(x)$ is balanced or constant. You'll see many variations of this algorithm online, but I think the simplest (and most elegant) one is as follows:

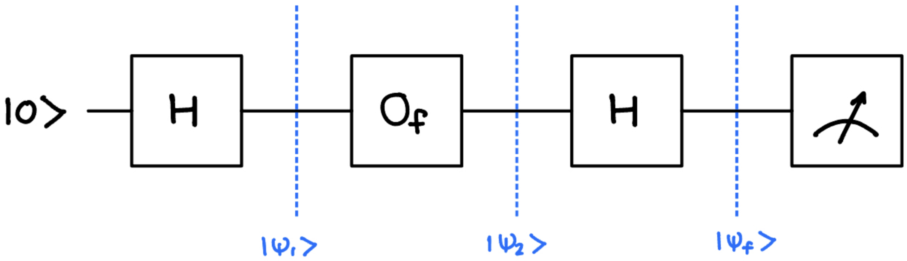


Figure 2: Deutsch's algorithm just involves 2 Hadamard gates, 1 query to the oracle, and a measurement at the end. We denote intermediate states in blue (like milestones in the algorithm) to make the calculation easier to follow.

We read quantum algorithms from left to right. So we start with our initial state $|0\rangle$, apply H, query the oracle (Equation 15), apply H again, and perform a measurement. This measurement should encode, in some way, the answer to our question: is $f(x)$ balanced or constant? Let's solve for the final state milestone by milestone ($|\psi_1\rangle \to |\psi_2\rangle \to |\psi_f\rangle$):

$$|\psi_1\rangle = H|0\rangle \tag{16}$$

$$= \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} \tag{17}$$

$$= \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} \tag{18}$$

$$
\begin{aligned}
|\psi_2\rangle &= O_f|\psi_1\rangle && (19)\\
&= \frac{1}{\sqrt{2}}O_f\begin{pmatrix}1\\1\end{pmatrix} && (20)\\
&= \frac{1}{\sqrt{2}}O_f(|0\rangle+|1\rangle) && (21)\\
&= \frac{1}{\sqrt{2}}(O_f|0\rangle+O_f|1\rangle) && (22)\\
&= \frac{1}{\sqrt{2}}\left[(-1)^{f(0)}|0\rangle+(-1)^{f(1)}|1\rangle\right] && (23)
\end{aligned}
$$

Note that we apply the definition of the oracle (Equation 15) to go from lines 22 to 23.

$$
\begin{aligned}
|\psi_f\rangle &= H|\psi_2\rangle && (24)\\
&= \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\1 & -1\end{pmatrix}\frac{1}{\sqrt{2}}\left[(-1)^{f(0)}|0\rangle+(-1)^{f(1)}|1\rangle\right] && (25)\\
&= \frac{1}{2}\left[\left((-1)^{f(0)}+(-1)^{f(1)}\right)|0\rangle+\left((-1)^{f(0)}-(-1)^{f(1)}|1\rangle\right)\right] && (26)
\end{aligned}
$$

Now we just have to measure our resulting qubit. In particular, we're after the probability of getting outcome $|0\rangle$; this is just the absolute-value-squared of the value in front of the $|0\rangle$ in Equation 26!

$$P_0 = \frac{1}{4}\left[(-1)^{f(0)}+(-1)^{f(1)}\right]^2 \tag{27}$$

Here's the punchline: If $P_0 = 0$, then $f(x)$ is ***balanced***. If $P_0 = 1$, then $f(x)$ is ***constant***. This might not be very obvious at first glance, so let's convince ourselves that this is true.

Suppose $f(x)$ is balanced; then $f(0) = 1, f(1) = 0$ OR $f(0) = 0, f(1) = 1$. Either way, we get

$$P_0 = \frac{1}{4}\left[(-1)^0+(-1)^1\right]^2 = 0 \tag{28}$$

What this means is that if $f(x)$ is balanced, we will **never** measure the qubit in the $|0\rangle$ state. It also implies that **if we do** measure the qubit in the $|0\rangle$, then $f(x)$ is constant. Just like that, we've finished our first quantum algorithm... congrats!