COS 597A: Quantum Cryptography — Princeton University
Lecture 3 (September 19, 2018)
Lecturer: Mark Zhandry — Scribe: James Bartusek

Notes for Lecture 3

1 Introduction

Recall from last time that our goal is to describe quantum systems with a mathematical model. We'll let $B = \{0,1\}^n$ be the usual set of classical states on n bits. A quantum state will be specified by $|\psi\rangle \in \mathbb{C}^{2^n}$ such that $||\psi\rangle|^2 = 1$, which assigns a complex amplitude to each classical state. For any $x \in B$, the state $|x\rangle$ puts an amplitude of 1 on x and 0 elsewhere. We refer to $\{|x\rangle\}_{x\in B}$ as the *computational basis*. Also recall from last time that our allowed operations on quantum states are unitary transformations U such that $U^\dagger U = I$, and measurements.

Remark: Last time, it was mentioned that phase changes have no realizable effect on a quantum state. This is partly true, *overall* phase changes do not have an effect, but partial phase changes can, as illustrated by the following examples. Let

$$x = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} \quad y = \frac{1}{\sqrt{2}}\begin{pmatrix}-1\\-1\end{pmatrix} \quad z = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}$$

Then y is obtained from x via an overall phase shift, but measuring both states give the same distribution. Furthermore, this change commutes with any unitary U, so applying any operation on these two states followed by a measurement gives the same distribution. On the other hand, z is obtained form x via a partial phase change. Consider applying the Hadamard gate $H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\1 & -1\end{pmatrix}$ to both x and z. We get that $Hx = \begin{pmatrix}1\\0\end{pmatrix}$ and $Hz = \begin{pmatrix}0\\1\end{pmatrix}$, which can be distinguished by measuring.

Questions:

- How do we implement unitary transformations and what is the computational complexity of these implementations? We'll talk about this next time.
- How can we use quantum systems to solve classical problems?

2 Reversible Computation

To answer the second question, we'll have to discuss the notion of reversible computation, which has been around for decades. To motivate the need for this, consider some classical function $f : \{0,1\}^n \to \{0,1\}^n$ that we would like to evaluate using a quantum system. Thus, we'll want to find some unitary transformation U such that $U|x\rangle = U|f(x)\rangle$. However, it can be shown that a matrix U is unitary if and only there does not exist $x \neq x'$ such that $f(x) = f(x')$. In other words, f has to be bijective for such a unitary U to exist. One direction of this follows from the fact that if U maps two different standard basis vectors to the same vector then it can't possibly be full rank and thus its inverse doesn't exist, a property required for it to be unitary. So, we'll need a way to transform every classical function into a bijective one, which is where reversible computation comes in.

It is certainly not true classically that every step of computation can be reversed. Consider for example the XOR gate $(a, b) \to (c = a \oplus b)$. Given c, it is impossible to recover a and b, so this gate actually loses information. This process of losing information has physical implications.

Landauer's Principle: Energy must be expended to lose information (and there is a particular conversion from amount of information lost to amount of energy that must be expended).

This gives a lower bound on the amount of energy that must be expended to compute some classical function f. Our current computers are very far from this lower bound, however, there has still been interest in devising reversible ways of computation that could theoretically circumvent this lower bound. For example, we can make the above XOR gate reversible as follows $(a, b) \to (a, c = x \oplus b)$. Now given a and c, we can XOR them to obtain b. Now what if we try to same technique of outputting one of the inputs for an AND gate, $(a, b) \to (a, c = a \wedge b)$. This is actually not reversible, since if $a = 0$ we have no idea what the bit b was. So we can try outputting both a, b along with their AND. This would work, but we'll tweak this to get something called the Toffoli gate, which has the nice property that it is its own inverse: $(a, b, c) \to (a, b, (a \wedge b) \oplus c)$. Whether or not the third input bit gets flipped tells us the result of the AND of the first two inputs.

Now how do we make a general function $f : \{0,1\}^n \to \{0,1\}$ reversible? We can follow the same technique as the Toffoli gate, and let the result of the function flip an auxiliary input. So we define $g(x, b) = (x, b \oplus f(x))$ where $x \in \{0,1\}^n$ and $b \in \{0,1\}$. It is easy to see that g is its own inverse and that it is bijective, so there are no more barriers to writing g as a unitary transformation.

However, we ideally would like this transformation from f to g to be efficient, and to be done in such a way that each step of the computation in the resulting g is reversible. We'll assume that we have an energy-preserving method of implementing the Toffoli gate, and we'll show how to carry out this transformation in an efficient way for arbitrary f.

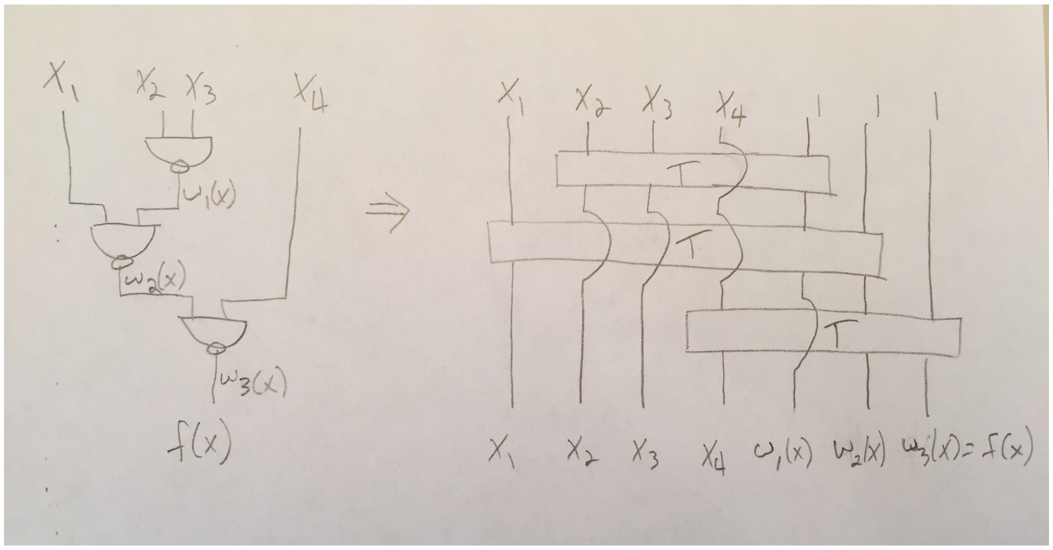
Proposition: Suppose $f : \{0,1\}^n \to \{0,1\}$ is implemented as a circuit of s NAND gates (recall that the NAND gate is universal for classical computation). Then g can be implemented using $\sim 2s$ Toffoli gates.

Proof. Let w_i be the i'th wire of the circuit computing f where w_s is the output wire, and let $w_i(x)$ be the value on the i'th wire on input x to the circuit. We'll define a function $h : \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^n \times \{0,1\}^s$ such that $h(x, 1^s) = (x, w_1(x), \dots, w_s(x))$. Let h' have the same domain and range and be such that $h'(x, w_1(s), ..., w_s(x)) = (x, 1^s)$. Assume for now that we have implemented both h and h' with Toffoli gates. We can set up the circuit below with $n + s + 1$ input and output wires each which computes $f(x)$ in a reversible manner.

$$(x, 1^s, b) \to \boxed{h(x, 1^s), b} \to (x, w_1(x), \dots, w_s(x), b) \to$$
$$\boxed{h'(x, w_1(x), \dots, w_s(x)), b \oplus w_s(x)} \to (x, 1^s, b \oplus f(x))$$

Note that we hardcode the second set of input wires to always be 1, and at the end of the computation they are all restored to 1 to be used for the next computation.

It remains to show how we implement h and h' with Toffoli gates. We'll show this by example, turning a circuit with 3 NAND gates into one with 3 Toffoli gates. Note that if the third input of a Toffoli gate is 1, then the third output is actually the NAND of the first two inputs. We'll use this fact to set up the circuit as follows, where the third input of each Toffoli gate is a fresh input wire hard-coded to 1 (T denotes a Toffoli gate in the picture below).



Now notice that since T is its own inverse, we can read the above circuit in the opposite direction to obtain an implementation for h'.

$\square$

3 Quantum Circuits

Now in order to compute any function $f : \{0,1\}^n \to \{0,1\}$, we can come up with some unitary U_f such that $U_f|x,b\rangle \to |g(x,b)\rangle = |x, b \oplus f(x)\rangle$. Indeed, since g is its own inverse, $U_f^2 = I$, and it is easy to see that U_f permutes the standard basis vectors, which together imply that U_f must be unitary. Now the question becomes how to implement arbitrary unitary transformations. Ideally, we will want to devise a set of small unitaries that can be stitched together in order to compute any arbitrary unitary, similar to how we define gates for classical circuits. Recall that if S is a finite set of functions called 'gates', a circuit consists of connecting these gates with wires under certain conditions (no loops, DAG structure, etc). We say that S is *universal* if for any function $f : \{0,1\}^n \to \{0,1\}$, there exists a circuit with gates in S that computes f. For example, $\{XOR\}$ is not a universal set of gates, but $\{NAND\}$, and $\{AND, NOT\}$ both are (which we won't bother proving in this class).

Now we consider the analogue of classical circuits in the quantum world and describe the main differences. To begin with, gates are now unitary transformations and wires contain qubits. Now we say that a finite set of unitaries S is universal if for every unitary U, there exists a circuit made up of unitaries from S that computes U. However, there is an issue with the above definition, since there are only countably many ways to stitch together unitaries from S into a circuit, yet there are uncountably many unitary transformations (consider for example all matrices $\begin{pmatrix} 1 & \\ & e^{i\Theta} \end{pmatrix}$ for all real $\Theta \in [0, 2\pi)$). We fix this by only requiring the set S to *approximately* implement any unitary (we won't get into the details here about what we mean precisely by approximate).

Finally, we discuss possibilities for universal quantum gate sets. We saw that the Toffoli gate was powerful enough to implement any reversible computation so might {Toffoli} be universal? It can't possibly be, because it is actually just a permutation matrix, so for example even the Hadamard transformation H cannot be performed. So what about {Toffoli, H}? The issue here is that these gates will map real vectors to only real vectors so how could phase shifts and other complex transformations be supported? It turns out that if we just add $\begin{pmatrix} 1 & \\ & e^{i\pi/4} \end{pmatrix}$, the set $\{\text{Toffoli}, H, \begin{pmatrix} 1 & \\ & e^{i\pi/4} \end{pmatrix}\}$ is now universal. A more typical example of a universal set is $\{CNOT, H, \begin{pmatrix} 1 & \\ & e^{i\pi/4} \end{pmatrix}\}$, where CNOT refers to the gate $(a, b) \to (a, a \oplus b)$.