6.896 Quantum Complexity Theory Sept 11, 2008

Lecture 3

Lecturer: Scott Aaronson *Scribe: Alex Arkhipov*

1. Last Time

1.1. **Entanglement.** A state is entangled if it cannot be decomposed into the product of one-qubit states.

1.2. **Bell's Inequality.** Sharing parts of a quantum state lets two players win a game with higher chance than achievable in the classical world (though it still won't let them send messages to each other faster than the speed of light). This shows that quantum physics is a non-local theory in which certain results cannot be explained by particles agreeing in advance.

1.3. **No-Cloning Theorem.** If we could make lots of copies of a quantum state and measure each of them, we could get lots of information about it. Unfortunately, we can't copy unknown quantum states, as a simple argument shows this would violate linearity.

2. Making Nature Work For Us

Last time we looked at quantum states with 2 qubits. Today, we jump up to n qubits.

An n qubit state is composed of 2^n state vectors, each with a complex amplitude.

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

This means that to work with 1000 particles, Nature must keep track of 2^{1000} complex numbers! That's more than there are atoms in the universe. As long as Nature is going through this exponential bookkeeping, why not put it to work to solve problems for us?

This is the idea of quantum computing, which was famously explored from two different angles by Richard Feynman and David Deutsch in the 1980's.

2.1. **Feynman.** In his 1982 lecture[1], Simulating Physics with Computers, Richard Feynman noted that having a computer do physics at a molecular level is hard because the work in simulating a quantum system grows exponentially in the number of particles. What if we level the field by simulating quantum physics on a computer made of quantum components? Even more, can we turn the tables and use such a quantum computer to efficiently solve problems that are classically hard?

2.2. **Deutsch.** David Deutsch was out to experimentally test the Many-Worlds Interpretation of quantum physics. Unlike the Bohr view, in which observers are classical and you can't even talk about them being in superposition, The Many-Worlds allows for a person to be in a superposition of possible worlds. Bohr's view raises questions about where the dividing line between quantum and classical line is: Atoms and molecules can be in superpositions, and the double-slit experiment has been done with Buckyballs, but where along the scale to human size is this superposability lost?

It doesn't seem practical to put our brains in superposition, so Deutsch suggested the next best thing: Make an artificially intelligent machine, put it in superposition, then ask it about its experiences.

[1]Simulating Physics with Computers `http://www.cs.berkeley.edu/~christos/classics/Feynman.pdf`

3. Complexity

Deutsch and Feynman had ideas for quantum computing, but as other 80's physicists, they missed the central issue of computation complexity: Is there some function that takes super-polynomial time to compute on a classical computer, but only polynomial time on a quantum computer?

The difficulty is that even if we have an exponential basis vector $\sum \alpha_x |x\rangle$, measuring it just gives a single random state $|x\rangle$ with probability α_x^2. So if the naive way doesn't work, how do we take advantage of this exponential vector? We exploit interference, choreographing the possible paths so that wrong answers cancel destructively, while correct ones reinforce by adding constructively. This is an intricate process that exploits the structure of the problem, so it's not clear we can do this for interesting problems.

4. Computation

So what exactly can a quantum computer do? This questions was answer in rigorous terms by Berstein and Vazirani in a 70 page paper[2] in 1993, who defined a quantum Turing machine which could have the tape head and symbols and superpositions.

Independently, Andy Yao defined quantum circuits, which are computationally equivalent. This is what people use today because they're simpler to think about.

So, in a quantum circuit, we start with N qubits, some of which contain the input to our problem, and the rest of which have been initialized to 0 and will serve as space for work. We apply some unitary transformation to the qubits, then measure them to get the answer.

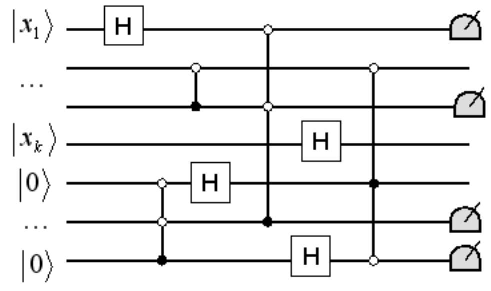

Figure 4.1. A quantum circuit

But which unitary transformation can we apply efficiently?

Maybe all of them? Let's return to classical logic circuits and look at a simple counting argument given by Claude Shannon.

How many Boolean functions are there on n bits, $f : \{0,1\}^n \to \{0,1\}$? The answer, 2^{2^n} is doubly exponential. And how many polynomially sized Boolean circuits are there? Only $2^{poly(n)}$. So, only a miniscule fraction of Boolean functions are computable by a circuit with a polynomial number of gates.

We want to only use gates that are *local*, meaning that they act on only 1 or 2 bits. We take a set of these gates as a basis, for example, $\{OR, AND, NOT\}$ or just $\{AND, NOT\}$, or even the single gate $\{NAND\}$. Each of these sets is universal, meaning that we can make any Boolean functions using some number of them.

The situation for quantum computers is similar. Physics are local, and interaction requires contact, so we want to work with 1 or 2, or perhaps 3, qubits at once. So, we want to make big unitary transformations as the product of gates which act on a small number of qubits and leave the rest unchanged. Is there some set of gates few-qubit gates that lets compose large transformations,

[2]Quantum Complexity Theory `http://puhep1.princeton.edu/~mcdonald/examples/QM/bernstein_siamjc_26_1411_97.pdf`

or at least approximate them? Deutsch, and Bernstein-Vazirani answered *yes* to this question, but the proof is hairy, so we won't go through it.

How many local gates do you need to simulate an arbitrary unitary transformation? Polynomial is definitely not enough, by a degrees-of-freedom counting argument: Each 2^n by 2^n matrix has about 2^{4n} degrees of freedom, while a 2-qubit unitary matrix has 4. So, we need an exponential number of gates. There are other arguments to show that you still need an exponential number to approximate.

For an upper bound, The Solovay-Kitaev Theorem, whose proof is also hairy, says that exponentially many gates are also enough.:

Theorem 4.1. *We can approximate an n-qubit unitary transformation to within ε error in L_2 using $O\left(2^n\left(n + polylog\frac{1}{\varepsilon}\right)\right)$ Hadamard and Toffoli gates.*

The gates they refer are an example of a universal let of gates. Let's look at some examples of these.

CNOT and $\pi/8$.

- CNOT:

$$|x, y\rangle \to |x, x \oplus y\rangle = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

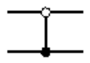

FIGURE 4.2. A CNOT gate

- Rotation by $\frac{\pi}{8}$: $\begin{bmatrix} \cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\ -\sin\frac{\pi}{8} & \cos\frac{\pi}{8} \end{bmatrix}$
- Complex scaling: $\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$. The preceding two gates are enough to approximate all real unitary matrices. With this, they can approximate complex ones as well.

CCNOT (Toffoli) and Hadamard.

- CCNOT (Toffoli): Perform a CNOT only if the first bit is 1, thus XOR'ing the AND of the first two bits onto the second.

$$|x, y, z\rangle \to |x, y, xy \oplus z\rangle$$

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

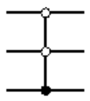

FIGURE 4.3. A CCNOT gate

- Hadamard: This gate switches back and forth from the standard $(|0\rangle, |1\rangle)$ basis and the Hadamard basis $\left(\frac{|0\rangle+|1\rangle}{\sqrt{2}}, \frac{|0\rangle-|1\rangle}{\sqrt{2}}\right)$

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

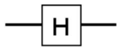

FIGURE 4.4. Hadamard gate

- Complex scaling: $\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$

Note that not every combination of small gates is universal. For example, with CNOT and Hadamard, one can only express a special subset of circuits, and this subset can be simulated efficiently by classical computers. This is the Gottesman-Knill Theorem.

5. Measurement

So we compose some polynomial number of these gates to make a circuit. We put through our inputs and some zeroes, and at the end perform a measurement. Since we're solving a decision problem, we'll just measure one qubit, which will tell us to accept or reject.

Could we gain more power by measuring things in the middle? After all, measurement can change the course of the computation.

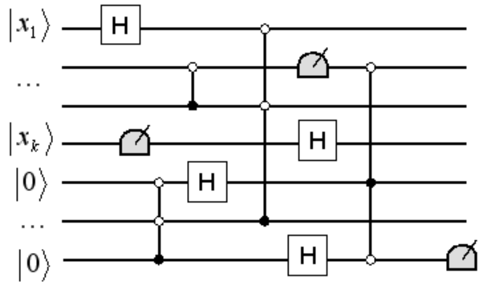

FIGURE 5.1. Do intermediate measurements help?

For example, applying two Hadamards takes $|0\rangle$ to $|0\rangle$, since it's the identity. But, if we put a measurement between them, the measurement collapse to either 0 or 1, making the state into $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ or $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$ after the second Hadamard, so we're equally likely to get a 0 or a 1.

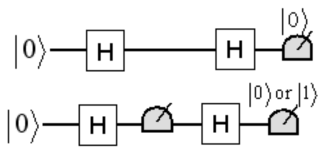

FIGURE 5.2. An intermediate measurement can change the result.

However, it turns our that measuring a qubit can be simulated by applying a CNOT from that qubit to a second qubit. We can call this the second qubit "measuring" the first one, since if we only look at the first qubit, this is indistinguishable from it actually being measured. So all intermediate measurements can be faked with unitary operations, and we can save actual measurements for the end. This is the Principle of Deferred Measurement.

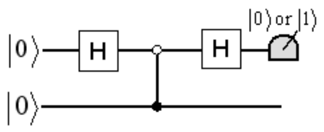

FIGURE 5.3. A CNOT with another qubit acts the same as a measurement

If we trace the evolution of the state through this circuit, we get:

$$|00\rangle \to \frac{|0\rangle + |1\rangle}{\sqrt{2}} |0\rangle \to \frac{|00\rangle + |11\rangle}{\sqrt{2}} \to \frac{|00\rangle + |01\rangle + |01\rangle - |11\rangle}{2}$$

Measuring the first qubit gives us equal chance of 0 or 1. Note how the effect of CNOT on the first qubit is the same as when we measured it, with the second qubit holding the result of the measurement.

The Principle of Deferred Measurement says something fundamental about the nature of measurement. This is why it's so hard to build a quantum computer - stray particles keep trying to CNOT your qubits.

6. Definition of BQP

We finally have all the tools to define BQP, the set of problems that can be solved in polynomial time with bounded error by a quantum computer. But first, let's recall the definitions of some classical complexity classes. For simplicity, we look at decision problems.

Definition. **P** is the class of languages $L \subset \{0,1\}^\star$ for which there exists a Turing machine M and a polynomial q so that for inputs $x \in \{0,1\}^n$, M terminate in at most $q(n)$ steps and accepts if and only if $x \in L$.

Definition. **PSPACE** is defined like P, except we're limited by $q(n)$ space, rather than time.

Definition. **EXP** is defined like P, but we're limited by $2^{q(n)}$ time steps.

Definition. **BPP** is the class of languages $L \subset \{0,1\}^\star$ for which there exists a Turing machine probabilistic M and a polynomial q so that for inputs $x \in \{0,1\}^n$, M terminate in at most $q(n)$ steps and

- If $x \in L$, then M accepts with probability $> 2/3$
- If $x \notin L$, then M accepts with probability $< 1/3$

The constants $1/3$ and $2/3$ aren't important; we can amplify to make the success probability as high as we want by running the program a bunch of times and taking the majority result. We can pull the randomness out of the definition of BPP by making M take an addition argument r, which can take values in $\{0,1\}^n$.

- If $x \in L$, then $M(x,r)$ accepts for at least $2/3$ the possible values of r
- If $x \notin L$, then $M(x,r)$ accepts for at most $1/3$ the possible values of r.

We can't pull randomness the same way out of BQP as we did for BPP. This is a key difference; randomness is intrinsic to quantum algorithms.

We have some inclusions:

- $P \subset BPP$. Why? Just don't use randomness.
- $BPP \subset PSPACE$. Why? Just run the problem for each possible r and keep a count of how many accepted.

- $PSPACE \subset EXP$. Why? Polynomial space can hold go through exponentially many configurations without looping.

We're ready to define BQP.

Definition. BQP is the class of languages $L \subset \{0,1\}^\star$ for which there exists a *uniform* family of polynomial-size quantum circuits $\{C_n\}$ over some basis of universal gates and a polynomial q so that for all n and inputs $x \in \{0,1\}^n$.

- If $x \in L$, then $C_n\left(|x\rangle\,|0\rangle^{\otimes q(n)}\right)$ accepts with probability $> 2/3$
- If $x \notin L$, then $C_n\left(|x\rangle\,|0\rangle^{\otimes q(n)}\right)$ accepts with probability $< 1/3$

Since circuits have to pre-specify the input size, so we need a circuit for each input size n. By uniform, we mean that there is a classically efficient algorithm to produce C_n given n.

Next time, we'll look at the properties of BQP and figure out where it lies within classical complexity classes.