6.896 Quantum Complexity Theory | Sept 4, 2008

Lecture 1

Lecturer: Scott Aaronson

1 Introduction

1.1 Course structure

Prof. Aaronson wants people to participate and show up. There will also be projects which could contain some original research, not necessarily earth-shattering. A literature survey would be fine as well.

There is no textbook. The syllabus recommends a few books on QC. Prof. Aaronson will try to put them on reserve in the library. There are also some great resources online, as mentioned in the syllabus.

The prerequisites are that you should know something about quantum computing *or* about computational complexity theory. The first few lectures may be boring to some, but Prof. Aaronson guarantees that no other lectures will be boring to anyone.

The central object that we will study is BQP (Bounded error Polynomial Time), a class of problems.

We will also study what is possible in QC in the so-called *black-box model*, the bestiary of quantum complexity classes (beyond BQP), and other miscellaneous topics.

1.2 Computability and Complexity Theory

What is complexity theory? It's the field that contains the $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ question.

Computability theory was established by Gödel, Turing, and others, who came up with a formal notion of computability, computability by a Turing machine. They showed that many problems are computable but that certain problems are not computable.

The question of complexity followed naturally. The theory of NP-completeness established the central question today: $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$.

What is an algorithm? It's a set of instructions for doing some computation. Since the 1930s, we've modeled it as instructions for a Turing machine.

The Church-Turing thesis states that—well, people disagree about what it should mean or even what Church and Turing themselves thought it meant. But one version is that everything that's computable in the physical world is computable by a Turing machine. And in particular, all programming languages are essentially equivalent—they can all simulate each other.

The Extended Church-Turing Thesis says that the time it takes to compute something on any one machine is polynomial in the time it takes on any other machine. This includes Turing machines, register machines, etc. Even before quantum computing, there were indications that the Extended Church-Turing Thesis might be on shakier ground than the original Church-Turing Thesis. For example, randomized algorithms are sometimes faster than deterministic algorithms, leading to a class called BPP (Bounded-Error Probabilistic Polynomial-Time). Obviously $\mathrm{P} \subseteq \mathrm{BPP}$. (Today we believe that $\mathrm{BPP} = \mathrm{P}$, but we can't prove it.)

NP (Nondeterministic Polynomial-Time) is the class of problems where, if the answer is "yes," then there exists a proof that's verifiable in polynomial time (though it might be very hard to find). An alternative definition is that a problem is in NP iff it's solvable in polynomial time by a Turing machine that can take branches, evaluate all paths simultaneously, and accept iff any of the paths would accept.

Our best understanding of physics corresponds to the class BQP (Bounded-Error Quantum Polynomial-Time). We're still trying to figure out exactly how it relates to the classical complexity classes.

2 Quantum mechanics

2.1 What is quantum mechanics?

Quantum mechanics is a theory in which several possibilities seem to happen at once. For example, a single photon given a choice between two slits, shows light and dark fringes, which seems to violate classical probability, in which one would assume that, when a second slit is open, the probability that the photon hits somewhere could only increase.

One question is what effect this has on computers and complexity: can we use quantum effects to compute things faster?

There are all kinds of interesting questions about how different concepts in complexity change with quantum mechanics. You can stick the work quantum in front of all kinds of things: communication channels, proofs, advice, pseudorandomness, etc.

We would like to organize our thoughts on these subjects.

2.2 Quantum mechanics as an extension of classical statistics

Many people today are under the bizarre misapprehension that quantum mechanics is hard. Prof. Aaronson blames the physicists. But, once you take the physics out, quantum mechanics is very easy – it's what you would inevitably get if you tried to generalize classical probability theory by allowing things that behaved like probabilities but could have minus signs.

The "state" of a classical computer could be written as a vector of all possible classical states, in which one element is 1 and the rest are 0. Operations are matrix multiplication. The downside of this representation is that it's exponentially inefficient, but the upside is that any operation can be seen as just a matrix multiplication.

One example of an elementary operation is controlled-NOT (CNOT)—it maps $00 \to 00$, $01 \to 01$, $10 \to 11$, and $11 \to 10$. In matrix language, this is

$$\text{CNOT} \equiv \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

In classical statistics, we change the vector to be a vector of probabilities—each element is nonnegative, and they all sum to 1. The matrices are now stochastic matrices (i.e. they conserve probability), and they can turn a deterministic state into a probabilistic one. (A stochastic matrix is one that is nonnegative, real, and for which each column sums to 1.)

Now we can ask: what happens if we decided to preserve the L_2 norm instead of the L_1 norm? We can also use complex numbers while we're at it. The picture is now a unit circle in which the

states (axes) might be named $|0\rangle$ and $|1\rangle$. (These are kets, a bizarre yet convenient notation due to Dirac.) At the end of the day, though, we want probabilities for the various possible outcomes ($|0\rangle$ or $|1\rangle$), and those probabilities should sum to one. The obvious choice is to take the squares of the amplitudes as our probabilities.

There are two things we can do to a computer: look at the result or advance the computation. Looking at the computation uses the probabilities above. Advancing the computation multiplies the vector of amplitudes by some norm-preserving matrix—i.e. a unitary matrix. Some examples are rotations and reflections.

As an example, suppose we apply the 45° rotation $\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$ twice. Classically, we'd expect that the first one randomizes the state and then the second one randomizes it again. But, in fact, it turns into a standard NOT gate. Another way to think of this is that all possible outcomes happen, but that $|0\rangle \to |0\rangle \to |1\rangle$ and $|0\rangle \to |0\rangle \to |1\rangle$ occur with opposite-signed amplitudes and cancel each other out. This is also what happens in the double-slit experiment.

Conservation of probability requires unitarity (i.e. the columns of the matrix should be orthogonal and have unit norm, or equivalently $A^{-1} = A^T$). This implies invertibility and thus all unitary evolutions are reversible. But looking at a quantum computer is *not* reversible—it collapses the wavefunction (state) into whatever you saw in the measurement, so the same measurement done twice in a row gives the same result.

6.845 Quantum Complexity Theory
Fall 2010