**6.841/18.405J: Advanced Complexity Theory** May 15, 2002

# Lecture 24

*Lecturer: Madhu Sudan* *Scribe: Johnny Chen*

## 1 Quantum Computing Wrap-Up

Recall that in the last lecture we saw Simon's algorithm and Shor's factoring algorithm. Today we consider some other issues and offer concluding remarks.

### 1.1 Grover's Algorithm

Suppose we have oracle access to a function $f : [N] \to \{0,1\}$ (here $[N] = \{1, \ldots, N\}$), and we want to know if there exists a $x$ such that $f(x) = 1$. This is a prototypical NP-complete problem, so we do not hope to solve in $O(\log N)$ time with a classical model of computation. But what about quantum computers? It turns out that a quantum algorithm needs $o(\sqrt{N})$ sequential queries to decide this problem [Bennett, Brassard, Bernstein, Vazirani]. Grover proved this bound was tight by giving an algorithm using $O(\sqrt{N})$ time, so it seems that $\text{NP}^f \subsetneq \text{BQF}^f$. The algorithms of Simon, Shor and Grover therefore demonstrate that quantum computing is definitely an interesting *model* to consider.

### 1.2 Problems and Pitfalls

It remains to be seen if a quantum computer can actually be built. Here we will consider some challenges.

**Error Correction**
Suppose we measure bits from a circuit by taking 0 to be -5V and 1 to be +5V. For small perturbations we can still take accurate measurements: for example, we take +4.9V to be a 1. However, these small perturbations accumulate, making the final measurements unreliable. In the classical model of computation two things save us and allow for error correction. First is the transistor, which in our example naturally pushes output close to $\pm 5V$. The second is a simple mechanism in which we encode 0s and 1s by repetition, so that 0 becomes $0 \cdots 0$ for example. As some of the 0s are corrupted, we can correct these errors by periodically considering the corrupted strings and changing them to $0 \cdots 0$ or $1 \cdots 1$ via majority vote. While this seems rather basic, it is crucial to classical computation, and not possible in quantum computation! We cannot measure periodically for self correction because the states will then collapse to a single one, destroying linear superposition. Also, since processes must be reversible, we cannot destroy the string.

Shor gave an elementary quantum error-correcting code that corrects errors without destroying or measuring the message. We will not discuss it, but it suffices to say results followed [AharanovBen-Or], [Shor], [Kitaev] that consider what it means to build a fault-tolerant quantum computer.

**Gates with Low Error**
One assumption permeating quantum computing is that quantum gates can be built with error probability at most $10^{-4}$. With this assumption we can assemble millions of such gates and perform real computations. As yet, no one has been able to build even one gate with this accuracy.

**Initializing and Resetting**
In our description of quantum computation model, we have assumed that we can reset the input to a gate to 0s (erasing the work tape). But how is this done in a quantum computer? It turns out that this is a nontrivial problem.

Physicists would say, it depends on the model. One particular model uses *Nuclear Magnetic Resonance* (NMR), where cooling the circuit has a "cleaning" effect. However, if we wish to have 0 with probability $1 - p$ and 1 with probability $p$, the cost of cooling grows *exponentially* with $\frac{1}{p}$. So to set $n$ bits to 0 requires

$\Omega(2^n)$ energy; this erases the advantage of quantum computing. A proposed solution was given by Schulman and Vazirani. They give an efficient procedure where circuits are cooled, and through preprocessing bits are separated into "very cold" (pure 0s) and "very hot" to maintain entropy.

**Other Work**
The areas of quantum communication, quantum information theory, quantum cryptography and quantum complexity are all quite active.

# 2 Complexity Wrap-Up

What have we learned this semester? Madhu's view falls into two broad categories.

## 2.1 Lower Bounds

Some techniques we have seen are:

- Diagonalization
- Circuit Complexity
- Communication Complexity

Admittedly, lower bounds were not discussed so much this semester, because complexity theorists have not been able to prove much (compared to the types of questions still open).

## 2.2 Identifying Computational Themes

We discussed some applications of Complexity theory.

- We abstracted some notions of games like Go, chess or Mahjongg and asked, for example, how does playing Go against a master compare to computer Mahjongg?
- A repeating theme was to identify a computational resource, define a complexity class and find a complete problem for this class.
- We explored some notions of proof, notably PCPs.

## 2.3 Topics Not Covered

**Proof Complexity: Resolution**
If we are given an unsatisfiable formula $\phi(x_1, \ldots, x_n)$, how can we prove that it is unsatisfiable? This is not an implausible task: consider the formula

$$\phi(x_1, x_2, x_3, x_4) = x_1 \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_3 \vee x_4) \wedge \overline{x_3} \wedge \overline{x_4}.$$

From the first two clauses, $x_2$. Combining this with the third clause gives $x_3 \vee x_4$. With the fourth clause, we have $x_4$, contradicting the last clause. Therefore $\phi$ is not satisfiable. Such an argument is called a *resolution derivation.*

Certainly we do not expect a short resolution proof on all instances, since this would imply co-NP=NP. An active area of research involves proving lower bounds for resolution. The first such papers appeared in the 80s [Haken85].

**Randomness**
Complexity Theory has been quite successful at answering questions relating to randomness. The main

consideration in this area is the difference between *actual randomness*, measured statistically, and *apparent randomness*, i.e., randomness required for computation.

Physicists believe randomness is everywhere. But how expensive is it? Consider three "worlds":

1. **Randomness is cheap**
   In this world, we can build randomized algorithms everywhere, and BPP rules.

2. **Randomness is "somewhat" expensive**
   In this world, randomness comes at the cost of polynomial slowdown. Therefore we need to reduce and recycle randomness. This area has had some nice results. For example, suppose that with a BPP algorithm $A$, $n$ random bits gives us error probability $1/4$. How many random bits do we need for error $1/16$? An immediate upper bound is $2n$, but it turns out that even with $A$ as a black box, $n+2$ random bits suffice. Indeed, for error $1/2^k$ we need $n + O(k)$ bits, where the constant has been pushed to $2 + \epsilon$ for some $\epsilon$.

3. **Randomness is infeasibly expensive**
   In this world, we must consider *pseudorandomness* - generating efficient objects with little or no randomness that are computationally indistinguishable from truly random objects. An example of a result in this area is the following

   **Theorem 1 (Impagliazzo, Widgerson)** *If every $f \in \boldsymbol{E} = DTIME(2^{O(n)})$ that has circuit complexity $2^{\Omega(n)}$ then there exists a pseudorandom generator ($\Rightarrow$ BPP=P).*

   A related question in this area is, what form does randomness take? We usually think of a random source as being a sequence of random, independent coin flips. What if the source is "dirtier"? Specifically, for a random variable $X$ on a set $\Omega$ define the min-entropy $H_\infty = \min_x \log_2 \Pr[X = x]^{-1}$. We will think of a source $X$ as "dirty" if we just know that $H_\infty = k$. The question becomes, can we produce pure random bits from a dirty source? It turns out that we cannot - but if use a purely random seed, we can. These objects are called extractors. One of the best extractor constructions uses $n$ bits from the source, a seed of length $\log n$ and outputs $m = \epsilon k$ purely random bits (again, $k$ is the min-entropy of the source). There turns out to be a fascinating connection between extractors and pseudorandom generators [Trevisan].

**Knowledge**

How do we formalize the notion of knowledge? Consider a message from Bob to Alice saying "I flipped a coin, and it landed heads". Intuitively, we know that Alice has received no knowledge from this statement: she has learned of the existence of a coin that can land heads up, a fact she could have verified herself. But when can we say knowledge has been exchanged in interaction? Indeed, the precise notion of knowledge was a barrier for those designing cryptographic protocols. The seminal work of Goldwasser, Micali and Rackoff [GMR] defined knowledge and zero-knowledge proofs (ZKP). The idea is that Alice has not learned anything from Bob's statement because she could have simulated it herself. While she may not be able to simulate it exactly, she can simulate some kind of distribution.