Computational Complexity October 22, 2018

Lecture 7 (Notes)

Lecturer: Ola Svensson *Scribes: Ola Svensson*

Disclaimer: These notes were written for the lecturer only and may contain inconsistent notation, typos, and they do not cite relevant works. They also contain extracts from the two main inspirations of this course:

1. The book *Computational Complexity: A Modern Approach* by Sanjeev Arora and Boaz Barak;
2. The course *http://theory.stanford.edu/ trevisan/cs254-14/index.html* by Luca Trevisan.

1 Introduction

Recall last lecture:

- Polynomial hierarchy (alternation of quantifiers)
  - $L \in \Sigma_i^p$ if exists polytime TM M and polynomial q such that
    $$x \in L \Leftrightarrow \exists u_1 \in \{0,1\}^{q(|x|)} \forall u_2 \in \{0,1\}^{q(|x|)} \cdots Q_i u_i \in \{0,1\}^{q(|x|)} M(x, u_1, \ldots, u_i) = 1 \qquad \forall x \in \{0,1\}^*,$$
  - $L \in \Pi_i^p$ if exists polytime TM M and polynomial q such that
    $$x \in L \Leftrightarrow \forall u_1 \in \{0,1\}^{q(|x|)} \exists u_2 \in \{0,1\}^{q(|x|)} \cdots Q_i u_i \in \{0,1\}^{q(|x|)} M(x, u_1, \ldots, u_i) = 1 \qquad \forall x \in \{0,1\}^*,$$
- Karp-Lipton Theorem: If $\mathbf{NP} \subseteq \mathbf{P}_{/\mathbf{poly}}$ then $\mathbf{PH} = \Sigma_2^p$.

Today:

- Why does circuit lower bounds seem so difficult? One answer is that any proof showing $\mathbf{NP} \subsetneq \mathbf{P}_{/\mathbf{poly}}$ cannot be a so called *natural proof.*
- Interactive proofs
  - **NP** can be defined as the class of languages which can be recognized by a polytime verifier given a proof/certificate (from a prover). This is an offline procedure.
  - What if the verifier and the prover can interact like we do in class?

2 Natural Proofs: a barrier for proving circuit lower bounds

There has been a large success in proving lower bounds in restricted models:

- Polysize circuits of unbounded fan-in for PARITY requires depth at least $O(\log n/\log\log n)$ (Håstad'86)
- CLIQUE does not have monotone circuits of polysize (Razborov'85)
- MATCHING does not have monotone circuits of polysize (Razborov'85)

(The latter result kind of destroyed the hope to use monotone circuits to separate **NP** from $\mathbf{P}_{/\mathbf{poly}}$.)

In addition Karp-Lipton's Theorem tells us that separating **NP** from $\mathbf{P}_{/\mathbf{poly}}$ seems like a good approach for the **P** vs **NP** problem.

However, our understanding of circuit lower bounds is embarrassing in general:

- Although almost all functions/languages require large circuits, we have been able to explicitly find any function that requires large circuits.
- In fact, it is open to find an explicit $f : \{0,1\}^n \to \{0,1\}$ that does not admit a $O(\log n)$ depth circuit of size $O(n)$.
- Another frontier is "Does **NEXP** have languages that require super-polynomial size circuits?". Or even "Does **NEXP** have languages that require $\omega(\log n)$ depth?".

In 1994, Razborov and Rudich described what they view as the main technical limitation of current approaches for proving circuit lower bounds:

- They defined the notion of "natural mathematical proofs" and pointed out that current lower bounds are (mostly) based on such proofs.
- They showed that obtaining strong lower bounds with such a proof would violate a stronger form of the $\mathbf{P} \neq \mathbf{NP}$ conjecture: namely that strong one-way functions do exist that cannot be inverted by algorithms running in subexponential time.
- Their result can be seen as a modern analog of the 1970s results on the limits of diagonalization (that we saw in class).

2.1 Definition of Natural Proofs

Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function and let $c \geq 1$ be some number. Any proof that f does not have a n^c-sized circuit can be viewed as exhibiting *some property* that f has, and which every function with n^c-sized circuits do not possess.

That is, such a proof can be seen as providing a predicate $\mathcal{P}$ on Boolean functions such that $\mathcal{P}(f) = 1$, but

$$\mathcal{P}(g) = 0 \qquad \text{for every } g \in \mathbf{SIZE}(n^c).$$

We call this condition n^c-usefulness as it rules out all the functions of circuits of size at most n^c.

So what are natural conditions on $\mathcal{P}$? Razborov and Rudich proposes two conditions:

Constructiveness: There is an $2^{O(n)}$-time algorithm that on input a function $g : \{0,1\}^n \to \{0,1\}$ outputs $\mathcal{P}(g)$. In other words, there is a polytime algorithm (in the size of the truth table of g) that evaluates $\mathcal{P}$.

Largeness: The probability that a random function $g : \{0,1\}^n \to \{0,1\}$ satisfies $\mathcal{P}(g) = 1$ is at least $1/n$.

Two examples of unnatural predicates:

1. $\mathcal{P}(g) = 1$ iff g is a Boolean function on n bits that has circuit complexity more than $n^{\log n}$. This predicate is clearly $c = o(n^{\log n})$ useful. Does $\mathcal{P}$ satisfy largeness? Yes almost all Boolean functions require exponential-sized circuits. However, we do not know if it satisfies constructiveness, since the trivial algorithm for computing it would be to enumerate all circuits of size $n^{\log n}$ which requires time $2^{n^{\log n}}$.
2. $\mathcal{P}(g) = 1$ iff g correctly solves the decision problem 3SAT on inputs of length n. This function is constructive: to compute it just enumerate all size n instances of 3SAT and check that g is correct on all instances. This takes $2^{O(n)}$ time. However, $\mathcal{P}$ does not satisfy the largeness property as it is 1 for exactly one function.

2.2 What's "Natural" about Natural Proofs?

The intuition of the constructiveness criteria is that most proofs use constructive properties. History tells us that it often is hard to exploit properties that do not have efficient algorithms. In fact many unconstructive arguments such as Lovasz Local Lemma and discrepancy was later constructivized. That said, there are of course unconstructive proof techniques (e.g. probabilistic method).

So what about largeness? Why should a lower bound for a specific function use a property that holds with good probability for a random function as well? The intuition is as follows

- Actually any proof that a function $f_0\{0,1\}^n \to \{0,1\}$ does not have a size S circuit, implies that at least half of the functions from $\{0,1\}^n$ to $\{0,1\}$ do not have a circuit of size $S/2 - 10$.
- To see this, choose a random $g : \{0,1\}^n \to \{0,1\}$ and observe that

  $$f_0 = (f_0 \oplus g) \oplus g,$$

  where $g \oplus h$ denotes the function that maps every input x to $g(x) \oplus h(x)$.
- Then we can see that if both $(f_0 \oplus g)$ and g have circuits of size $< S/2 - 10$ then f_0 has a circuit of size $< S$.
- Since both g and $(f_0 \oplus g)$ are uniformly distributed, it follows that at least half of the functions must have circuits of size at least $S/2 - 10$.

2.3 Statement and explanation of result

Theorem 1 *Suppose that subexponentially strong one-way functions exist. Then there exists a constant $c \in \mathbb{N}$ such that the is no n^c-useful natural predicate.*

The assumption of the theorem is basically as follows: There are polytime computable functions $f : \{0,1\}^* \to \{0,1\}$ such that any algorithm, that given y (with good probability) finds an x such that $f(x) = y$ needs time 2^{n^ϵ} for some $\epsilon > 0$. We believe that such functions exist (think crypt e.g. factoring).

- Note that theorem implies that if subexponentially strong one-way functions exist, then no natural proof can prove $\mathbf{NP} \subsetneq \mathbf{P}_{/\mathbf{poly}}$.

We do not cover the proof in this part of the course but it is given with (much) more interesting information in Chapter 23 of the textbook.

3 Interactive Proofs

- Standard notion of a mathematical proof is closely related to the certificate definition of **NP**:

  > To prove that a statement is true one provides a sequence of steps/symbols, and the verifier checks that they present a valid proof/certificate.

- However, people often use a more general way to convince one another of the validity of statements:

  > They *interact* with one another, where the person verifying the proof (called *verifier*) asks the person providing it (the *prover*) for a series of explanations before he is convinced.

- Interactive proofs aim to understand such proofs from a complexity-theoretic perspective.
- We will see that if the verifier has to be deterministic then we can still only recognize **NP**, i.e., interaction does not add any power.

- However, if we add randomness to the verifier, then we get **PSPACE**!
- When adding randomness to the verifier one can distinguish between *private* vs *public* coins. We will discuss this difference in this lecture and prove the **PSPACE** result in the next lecture.

3.1 Warm up: Interactive Proofs with deterministic verifier (and prover)

When defining interactive proofs, it makes sense to allow the prover any computational power (after all, he or she can have proved something really great) but it should be easy to verify. Therefore, we will always require the verifier to run in polynomial time (but have no restrictions on the prover)!

Let us now formally define "interaction".

Definition 2 *Let $f, g : \{0,1\}^* \to \{0,1\}^*$ be functions and k an integer (allowed to depend on input size). A k-round interaction of f and g on input $x \in \{0,1\}^*$, denoted by $\langle f, g\rangle(x)$ is the sequence of strings $a_1, \ldots, a_k$ defined as follows:*

$$
\begin{aligned}
a_1 &= f(x) && \textit{(verifier computes first question based on input)}\\
a_2 &= g(x, a_1) && \textit{(prover answers first question)}\\
a_3 &= f(x, a_1, a_2) && \textit{(verifier computes first question based on input and previous answers)}\\
a_4 &= g(x, a_1, a_2, a_3) && \textit{(prover answers second question)}\\
&\vdots\\
a_{2i+1} &= f(x, a_1, \ldots, a_{2i}) && \textit{for } 2i < k\\
a_{2i+2} &= g(x, a_1, \ldots, a_{2i+1}) && \textit{for } 2i+1 < k
\end{aligned}
$$

The output *of f at the end of the interaction denoted $out_f\langle f, g\rangle(x)$ is defined to be $f(x, a_1, \ldots, a_k)$; we assume this output is in $\{0,1\}$.*

Having defined interaction, the definition of a deterministic proof system follows quite naturally.

Definition 3 *We say that a language L has a k-*round deterministic proof system *if there is a deterministic TM V that on input $x, a_1, \ldots, a_i$ runs in time polynomial in $|x|$, and can have a k-round interaction with any function P such that*

(Completeness) $x \in L \Rightarrow \exists P : \{0,1\}^* \to \{0,1\}^* : out_V\langle V, P\rangle(x) = 1.$

(Soundness) $x \notin L \Rightarrow \forall P : \{0,1\}^* \to \{0,1\}^* : out_V\langle V, P\rangle(x) = 0.$

The class **dIP** *contains all languages with a $k(n)$-round deterministic interactive proof system where $k(n)$ is polynomial in $n = |x|$.*

3.2 dIP = NP and a simple randomized protocol

It is an exercise to see that deterministic interaction does not add any power:

Exercise 1 *We have* **dIP** = **NP**.

In light of the above, we extend the power of the verifier by allowing him to use randomization. We now give a simple example that illustrates the benefit of randomness in this setting. Consider the following problem:

- Marla (prover) has one red sock and one yellow sock, but her friend Arthur (verifier), who is color-blind, does not believe her that the socks have different colors. How can she convince him that this is really the case?

We assume that Arthur takes the two socks from Marla, and he can then flip coins and then ask Marla questions based on the outcome of these random tosses. How can he do it so as to get convinced (with reasonable probability) that Marla has two different socks if it is the case?

The protocol is as follows:

- Marla gives both socks to Artur. Artur holds one of the socks in each hand.
- Marla then turns her back to Artur and he tosses a coin.
- If then coin flip is "heads" then he keeps the socks as they are and otherwise he switches them between his left and right hands.
- He then asks Marla to guess whether he switched or not.

We leave as an exercise to prove that, if the socks have different colors, then Marla can always answer correctly. Otherwise, if they have the same color, she can answer correctly with probability at most 1/2. Repeating the protocol, say 100 times, improves the error guarantee and so Artur can be sure whether the socks are of the same color or not.