Computational Complexity October 25, 2018

Lecture 8 (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Disclaimer: These notes were written for the lecturer only and may contain inconsistent notation, typos, and they do not cite relevant works.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:

- Natural proofs (Razborov and Rudic'94): a barrier for proving lower bounds on circuits. If the predicate $\mathcal{P}$ distinguishing your function from those of small circuits (polysize) satisfies

  Constructiveness: $\mathcal{P}(g)$ can be evaluated in time $2^{O(n)}$ for any $g : \{0,1\}^n \to \{0,1\}$;
  Largeness: $\Pr_g[\mathcal{P}(g) = 1] \geq 1/n$

  then no subexponentially strong one-way functions exist (a conclusion we believe is unlikely).

- Interactive proofs aim to understand interactive proofs from a complexity-theoretic perspective.
- Interaction does not add any power if it is deterministic, i.e., **dIP** = **NP**.

Today:

- Today we see that randomness (seemingly) adds power to the verifier. In particular, we consider graph nonisomorphism (GNI) and show that if the verifier is allowed to use randomness, then GNI $\in$ **IP**. In contrast, GNI is not known to be in **NP**.
- We also distinguished between private random coins (**IP**) and public coins where the prover sees the randomness of the prover (**AM**).
- Surprisingly, public and private coin models are not very different:

$$\mathbf{AM}[k+2] \subseteq \mathbf{IP}[k].$$

2 The class IP: Interactive Proofs with a Probabilistic Verifier

We want to capture the additional power of a probabilistic verifier seen in the socks example from last lecture. To this end, we extend "deterministic" interaction to that of including a probabilistic verifier:

- The verifier f is now probabilistic, so we add an additional m-bit input r to the function f.
- That is the interaction is now

$$\begin{aligned} a_1 &= f(x, r) \\ a_2 &= g(x, a_1) \\ a_3 &= f(x, r, a_1, a_2) \end{aligned}$$

  and so on.

- Notice that the prover (g) does *not* see the random coins. For this reason this is called *private coins* as opposed to *public coins* that we discuss later.
- The interaction $\langle f, g\rangle(x)$ is now a random variable of $r \in \{0,1\}^m$. Similarly the output $\text{out}_f\langle f, g\rangle(x)$ is also a random variable of r.

Definition 1 (Probabilistic verifiers and the class IP) *For an integer $k \geq 1$, we say that a language L is in* $\mathbf{IP}[k]$ *if there is a probabilistic TM V that on input $x, r, a_1, \ldots, a_i$ runs in time polynomial in $|x|$, and can have a k-round interaction with any function P such that*

(Completeness) $x \in L \Rightarrow \exists P : \{0,1\}^* \to \{0,1\}^* : \Pr[\mathit{out}_V\langle V, P\rangle(x) = 1] \geq 2/3.$

(Soundness) $x \notin L \Rightarrow \forall P : \{0,1\}^* \to \{0,1\}^* : \Pr[\mathit{out}_V\langle V, P\rangle(x) = 1] \leq 1/3.$

We define $\mathbf{IP} = \cup_{c\geq 1}\mathbf{IP}[n^c]$.

Remark We give the following remarks:

- The constants 2/3 and 1/3 are arbitrary and they can be made very small (just as for **BPP**) using sequential repetition that repeats the basic protocol m times and takes the majority answer. In fact, using a more complicated proof, it can be shown that we can decrease the probability of error without increasing the number of rounds using something called parallel repetition (where we run the m executions of the protocol in parallel).
- Replacing the constant 2/3 with 1 in the completeness requirement does not change the class **IP**. (This is a non-trivial fact.)
- In contrast, replacing the constant 1/3 with 0 in the soundness case is equivalent to having a deterministic verifier and hence reduces the class **IP** to **NP**.

2.1 Interactive proof for graph nonisomorphism

We give an example of a language in **IP** that is not known to be in **NP**.

- Two graphs G_1 and G_2 are *isomorphic* if they are the same up to a renumbering of vertices; in other words, if there is a permutation π such that $\pi(G_1) = G_2$.
- The GI problem is the following: given G_1, G_2 decide if they are isomorphic.
- Clearly GI is in **NP**. Why? It is actually not known whether GI is **NP**-complete or in **P**. Together with factoring, it is the most famous such unresolved problem. Recently, Laszlo Babai made major progress by showing that GI can be solved in time $O(n^{\log n})$.

Here we give an interactive proof for the complementary problem GNI that is not known to be in **NP**. That is GNI is the following problem: decide whether two given graphs G_1, G_2 are *not* isomorphic.

Protocol: Private-coin Graph Nonisomorphism (The protocol is very similar to the "sock"-protocol.)

V: Pick $i \in \{1,2\}$ uniformly at random. Randomly permute the vertices of G_i to get a new graph H. Send H to P.

P: Identify which of G_1, G_2 was used to produce H. Let G_j be that graph. Send j to V.

V: Accept if $i = j$; reject otherwise.

As in the "sock"-protocol, it is easy to see that if the graphs are not isomorphic then the verifier always accepts. However, if they are isomorphic, the verifier accepts with probability $\leq 1/2$ (repeating the protocol decreases this probability further).

Remark One can see that in the above protocol the prover reveals no information except that the graphs are nonisomorphic. This is called a zero-knowledge proof because the verifier learns nothing except the truth of the statement. This has been formalized and studied. It has wide spread applications. For example, to prove that you hold the password without revealing the password itself.

3 Public Coins and AM

Our proof system for the "socks" problem and graph nonisomorphism seemed to crucially rely on the verifier's access to a source of *private* random coins that are not seen by the prover. Allowing the prover full access to the verifier's random string leads to the model of *interactive proofs with public coins*

Definition 2 (AM) *For every k, the complexity class $\mathbf{AM}[k]$ is defined as the subset of $\mathbf{IP}[k]$ obtained when we restrict the verifier's messages to be random bits and not allowing it to use any other random bits that are not contained in these messages.*

- An interactive proof where the verifier has this form is called a *public coin* proof, sometimes also known as an Arthur-Merlin proof.
- Note that we are not assuming that the verifier sends any other messages than the random coins. This is w.l.o.g. as the almighty prover can reconstruct everything by seeing the random coins of the verifier.
- Also note that the verifier does not get to see all random coins at once but rather they are revealed to the prover iteratively message by message.
- We denote (slightly inconsistently) by $\mathbf{AM}$ the class $\mathbf{AM}[2]$. That is $\mathbf{AM}$ is the class of languages with an interactive proof that consists of the verifier sending a random string, and the prover responding with a message, where the verifier's decision is obtained by applying a deterministic polynomial-time function to the transcript.

3.1 Simulating Private Coins by Public Coins

Clearly, for every k, $\mathbf{AM}[k] \subseteq \mathbf{IP}[k]$. More surprisingly:

Theorem 3 (Goldwasser-Sipser'87) *For every $k : \mathbb{N} \to \mathbb{N}$ with $k(n)$ computable in poly(n),*

$$\mathbf{IP}[k] \subseteq \mathbf{AM}[k+2].$$

Instead of proving this theorem in its full generality we prove the also surprising fact:

Theorem 4 GNI $\in \mathbf{AM}[2]$

The proof of this theorem demonstrates the power of recasting the problem. Indeed, it seems very hard to simply adapt the protocol with private coins that we saw previously. Instead, look at graph nonisomorphism in the following more quantitative way:

- Consider the following set of labeled graphs

$$S = \{H : H \text{ is isomorphic to } G_1 \text{ or } G_2\}.$$

Note that it is easy to certify that a graph H is a member of S by providing the permutation mapping of either G_1 or G_2 to H.

- An n-vertex graph has at most $n!$ equivalent graphs (all permutations). For simplicity, assume that both G_1 and G_2 have each exactly $n!$ equivalent graphs (symmetries can be dealt with appropriately).

- Then, we have
  - If G_1 is not isomorphic to G_2, $|S| = 2n!$ (and $|S \times S \times S \times S| = 16(n!)^4$).
  - If G_1 is isomorphic to G_2, $|S| = n!$ (and $|S \times S \times S \times S| = (n!)^4$).

- Hence, if we let $S' = S \times S \times S \times S$, we have reduced the problem of the prover to that of convincing the verifier that $|S'| \geq 16(n!)^4$ holds instead of $|S'| \leq (n!)^4$.

- This is done by a *set lower bound protocol* that we now describe.

Set Lower Bound Protocol (Simplified version of Goldwasser-Sipser protocol):

Conditions: S is a set such that membership in S can be certified. Both partities know a number K. The prover's goal is to convince the verifier that $|S| \geq K$ and the verifier should reject with good probability if $|S| \leq K/16$.

V: Randomly pick a hash function $h : S \to \{1, 2, \ldots, K/4\}$ (that sends each element in S to a random element in $\{1, \ldots, K/4\}$. Pick $y \in \{1, \ldots, K/4\}$ uniformly at random. Send h, y (or equivalently the randomly used bits) to the prover.

P: Try to find an $x \in S$ such that $h(x) = y$. Send such an x to V together with a certificate that $x \in S$.

V's output: If $h(x) = y$ and the certificate validates that $x \in S$ then accept; otherwise reject.

Remark The hash function as written now would take too many random bits to encode, namely $|S| \log(K/4)$. However, one can prove that a family of pairwise independent hash function suffices (not too hard, see textbook). Such a function can then be sent using $O(\log |S|)$ random bits which is polynomial.

Let us now analyze the protocol. Clearly, the prover (being all powerful) can make the verifier accept iff h, y happen to be such that $x \in S$ exists satisfying $h(x) = y$.

Claim 5 *The protocol has completeness at least* $2/3$*, i.e., if* $|S| \geq K$ *then the verifier outputs* 1 *with probability at least* $2/3$*.*

Proof

- Let y be the randomly chosen element in $\{1, \ldots, K/4\}$.

- The probability that h hashes an element x from S to y is

$$1 - \Pr[h(x) \neq y \text{ for all } x \in S] = 1 - \left(1 - \frac{4}{K}\right)^K \geq 2/3.$$

(For the equality we used that the hash function sends each element in S to a random element in $\{1, \ldots, K/4\}$ independently and uniformly at random.)

■

Claim 6 *The protocol has soundness at most* $1/3$*, i.e., if* $|S| \leq K/16$*, then the verifier outputs* 1 *with probability at most* $1/3$*.*

Proof

- Let y be the randomly chosen element in $\{1, \ldots, K/4\}$.
- The elements in S are at most hashed to $K/16$ elements in $\{1, \ldots, K/4\}$, i.e., a $1/4$ fraction of the elements.
- Since the hashing is uniform, the probability that there exists $x \in S$ such that $h(x) = y$ is at most $1/4$.

■

This finishes the analysis of the set lower bound protocol and thereby also of the fact that GNI $\in$ **AM**.

Some remarks:

- Theorem 3 is a bit harder but similar: in that case the idea is to make the set S contain all the (private) random strings that makes the private-coin verifier accept.
- Our protocol for set lower bound does not have perfect completeness. However, this can be achieved and thus implying a perfectly complete public-coins proof for GNI.
- Moreover, (as almost always) the soundness can be made exponentially small in the size of the input.

4 Evidence that Graph Isomorphism is *not* NP-complete

Recall our result on GNI: (after slight modifications to obtain perfect completeness and error reduction), there exists a probabilistic public-coin verifier V that asks a single question to the verifier P such that

$$\langle G_1, G_2 \rangle \in \text{GNI} \Rightarrow \exists P : \Pr[V \text{ accepts}] = 1$$
$$\langle G_1, G_2 \rangle \notin \text{GNI} \Rightarrow \forall P : \Pr[V \text{ accepts}] < \frac{1}{2^{n+1}}$$

Based on this results, we can show the following (proved by Boppana, Håstad, Zachos'87):

Theorem 7 *If* GI *is* **NP***-complete, then* $\Sigma_2^p = \Pi_2^p$ *(in other words, the polynomial hierarchy collapses to its second level).*

Proof

- It is sufficient to show that the assumption implies $\Sigma_2^p \subseteq \Pi_2^p$ as Σ_2^p is the complement of Π_2^p. Hence, it implies that $\Sigma_2^p = \Pi_2^p$.
- If GI is **NP**-complete, then GNI is **coNP**-complete.
- This implies that there is a function (reduction) f such that for every n variable formula φ

$$\forall y \in \{0,1\}^n \varphi(y) \text{ holds } \Leftrightarrow f(\varphi) \in \text{GNI}.$$

- Consider an arbitrary Σ_2^pSAT formula $\psi = \exists x \in \{0,1\}^n \forall y \in \{0,1\}^n : \varphi(x,y)$.
- The formula ψ is equivalent to $\exists x \in \{0,1\}^n g(x) \in \text{GNI}$, where $g(x) := f(\varphi(x,\cdot))$, i.e., the formula obtained from φ by fixing x.
- Now using our results about Arthur-Merlin proofs for GNI we have that GNI has a two round **AM** proof with perfect completeness and soundness error less than $2^{-(n+1)}$.
- Let V be the polytime verifier for this proof system and denote by m the length of the verifier's random tape and by m' the length of the prover's message.
- We claim that ψ is true if and only if

$$\forall r \in \{0,1\}^m \exists x \in \{0,1\}^n \exists a \in \{0,1\}^{m'} : V(g(x), r, a) = 1. \tag{1}$$

- Indeed, if ψ is true, then perfect completeness clearly implies the above.
- On the other hand, if ψ is false, this means that

$$\forall x \in \{0,1\}^n g(x) \notin \text{GNI}.$$

- Now, using the fact that the soundness error of the interactive proof is less than $2^{-(n+1)}$ and the number of x's is 2^n, we conclude that there exists a string $r \in \{0,1\}^m$ such that for every $x \in \{0,1\}^n$, the prover in the **AM** proof has no response a that will cause the verifier to accept. In other words, (1) is false in this case as required.
- Since deciding the truth of (1) is in Π_2^p, we have shown $\Sigma_2^p \subseteq \Pi_2^p$.

■