Topics in Theoretical Computer Science April 11, 2016

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Disclaimer: These notes were written for the lecturer only and may contain inconsistent notation, typos, and they do not cite relevant works.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

- 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**.
- However, if the verifier was allowed to use randomness, then GNI $\in$ **IP** and GNI is not known to be **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:

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

Today:

- We will show that **IP** = **PSPACE**, i.e., **IP** is believed to be a much larger class than **NP**.
- However, we start with a consequence of last lecture: GI is unlikely to be **NP**-complete.

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

Recall the result of the last lecture: GNI $\in$ **AM**[2]. In other words (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 1 *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 $\Sigma_2^p \subseteq \Pi_2^p$.

■

3 IP = PSPACE

- In the late 80's it was an open question to characterize **IP**. All we knew was that **NP** $\subseteq$ **IP** $\subseteq$ **PSPACE**.
- There were evidence that the first inclusion was proper (e.g., GNI $\in$ **IP**) and most researchers felt that the second containment would also be proper (as, in many settings, randomness do not seem to add too much power).
- However, that intuition was all wrong as the following result shows.

Theorem 2 IP = PSPACE

The theorem was proved in 1990 by Adi Shamir following (closely) work by Lund, Fortnow, Karloff, and Nisan appearing the same year.

To illustrate the main technique (arithmetization), we first give a proof that $\#SAT \in \mathbf{IP}$ and we then show that $PSPACE \subseteq \mathbf{IP}$. That $\mathbf{IP} \subseteq \mathbf{PSPACE}$ is left as an exercise.

3.1 $\#SAT \in \mathbf{IP}$

Define

$$\#SAT = \{\langle \varphi, K \rangle : \varphi \text{ is a 3CNF formula and it has } \textit{exactly } K \text{ satisfying assignments}\}.$$

Notice that $\#SAT$ is a pretty difficult language: it contains $\overline{\text{3SAT}}$ as a special case when $K = 0$.

To define an interactive proof for $\#SAT$ the main idea is arithmetization:

Definition 3 (Arithmetization) *Given a boolean formula $\varphi(x_1, \ldots, x_n)$, define a polynomial $g(x_1, \ldots, x_n)$ such that*

$$\varphi(x) = g(x) \qquad \forall x \in \{0,1\}^n.$$

How can we arithmetize a given 3CNF formula φ?

- Note that $0, 1$ can be thought of both as truth values and elements of some finite field.
- So if we consider φ consisting of n variables $x_1, \ldots, x_n$ and m clauses, we introduce field variables $X_1, \ldots, X_n$.
- Furthermore, for any clause of size 3, we can write an equivalent degree 3 polynomial as in the following example:

$$x_1 \vee \bar{x}_2 \vee x_3 \leftrightarrow X_i(1 - X_j)X_k.$$

- Let us denote the polynomial of the jth clause by $p_j(X_1, \ldots, X_n)$. For every $0, 1$ assignment we have $p_j(X_1, \ldots, X_n) = 1$ iff the assignment satisfies the clause.
- Multiplying these polynomials we obtain a multivariate polynomial

$$P_\varphi(X_1, \ldots, X_n) = \prod_{j=1}^{m} p_j(X_1, \ldots, X_n)$$

that evaluates to 1 on satisfying assignments and to 0 for unsatisfying assignment.

Observation 4 *The polynomial P_φ satisfies the following:*

1. *It has degree at most $3m$.*
2. *It has a representation of size $O(m)$ (just keep the p_j's) and given, $X_1, \ldots, X_n$, P_φ can be evaluated in polynomial time.*

Theorem 5 $\#SAT \in \mathbf{IP}$

Proof

- Given $\langle \varphi, K \rangle$, the number of satisfying assignments $\#\varphi$ of φ satisfies

$$\#\varphi = \sum_{b_1 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} P_\varphi(b_1, \ldots, b_n). \tag{2}$$

- Hence, the verifier's task is to verify that (2) equals K.
- First the prover sends the verifier a prime p in the interval $(2^n, 2^{2n}]$. Then the verifier checks that p is a prime (in polynomial time).
- Note that since (2) is between 0 and 2^n, this equations is true over the integers iff it is true over modulo p. Thus from now on we consider (2) as an equation in the field $\mathbb{F}_p$.
- We finish the proof of the theorem by showing a general protocol, *Sumcheck*, for verifying equations such as $(2) = K$.

Sumcheck protocol

Input: A degree d polynomial $g(X_1, \ldots, X_n)$, an integer K, and a prime p. The polynomial g is assumed to have a $poly(n)$-sized representation, degree $d \leq poly(n)$, and that it can be evaluated (given the input) in polynomial time. (Note that P_φ satisfies these conditions.)

Task: Design an interactive proof for the claim

$$K = \sum_{b_1 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(b_1, \ldots, b_n) \tag{3}$$

The protocol is as follows:

V: If $n = 1$, check that $g(1) + g(0) = K$. If so accept; otherwise reject. If $n \geq 2$ proceed as follows:

P: Sends $s(X_1)$ that is supposed to equal $h(X_1) := \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(X_1, b_2 \ldots, b_n)$.

(Note that $s(X_1)$ is a univariate polynomial of degree $\leq d$ if g has degree d and can therefore be transmitted efficiently.)

V: Reject if $s(0) + s(1) \neq K$; otherwise pick a *random* number $a \in \{0, \ldots, p-1\}$. Recursively use the same protocol to check that

$$s(a) = \sum_{b_2 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} g(a, b_2 \ldots, b_n) \qquad (= h(a))$$

Clearly, the interactive proof has perfect completeness, i.e., if (3) is satisfied then if the prover always answers as intended the verifier will always accept. We continue to analyze the soundness:

Claim 6 *If* (3) *is false, then V rejects with probability at least* $\left(1 - \frac{d}{p}\right)^n$.

Proof The proof is by induction on n.

Base case: For $n = 1$, V simply evaluates $g(0)$ and $g(1)$ and rejects with probability 1 if their sum is not K. So in this case we have perfect soundness.

Inductive step:

- If the prover returns the polynomial $h(X_1)$ then the verifier rejects as $h(0) + h(1) \neq K$.
- So assume that the prover returns $s(X_1)$ different from $h(X_1)$.
- As the degree d nonzero polynomial $s(X_1) - h(X_1)$ has at most d roots, there are at most d values such that $s(a) = h(a)$.
- Thus when V picks a random a,

$$\Pr_a[s(a) \neq h(a)] \geq 1 - \frac{d}{p}.$$

- If $s(a) \neq h(a)$ then the prover is left with an incorrect claim to prove in the recursive step (with one less variable). By the IH, the prover fails to prove this false claim with probability at least $\left(1 - \frac{d}{p}\right)^{n-1}$.
- Thus,

$$\Pr[V \text{ rejects}] \geq \left(1 - \frac{d}{p}\right)\left(1 - \frac{d}{p}\right)^{n-1} = \left(1 - \frac{d}{p}\right)^{n},$$

  which completes the inductive step and the proof of the claim.

■

The proof of Theorem 5 follows from the above claim as in that case $\left(1 - \frac{d}{p}\right)^n \geq \left(1 - \frac{poly(n)}{2^n}\right)^n$ which is very close to 1. ■

4 Exercises

Exercise 1 *In the interactive protocol for #SAT the prover sent a prime p to the verifier. Why don't you think the verifier choose the prime?*

Exercise 2 *To prove that* **PSPACE** $\subseteq$ **IP** *we shall use that the following problem is* **PSPACE***-complete:*

$$\text{TQBF} = \{\Psi : \Psi \textit{ is a quantified } \text{true} \textit{ Boolean formula of the form } \forall x_1, \exists x_2, \ldots, Q_n x_n \varphi(x_1, \ldots, x_n)\}.$$

Note that a formula $\Psi = \forall x_1, \exists x_2, \ldots, Q_n x_n \varphi(x_1, \ldots, x_n)$ *can be arithmetized and* $\Psi \in \text{TQBF}$ *if and only if*

$$\prod_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \prod_{b_3 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} P_\varphi(b_1, \ldots, b_n) > 0.$$

So it looks like we can run the following protocol. The prover sends a number $K > 0$ *and we run the sumset protocol to check that the LHS of the above equation equals* K*.*

There are two problems with this (one minor and one slightly bigger). What are they?

Exercise 3 *Prove that* **IP** $\subseteq$ **PSPACE***. Recall that* **PSPACE** *contain those languages recognizable by a TM that uses polynomial amount of space.*

5 PSPACE $\subseteq$ IP

In this section we give an interactive proof of the **PSPACE** complete problem TQBF. As any other $L \in$ **PSPACE** is polytime reducible to TQBF, it follows that **PSPACE** $\subseteq$ **IP**. Rather than reexplaining the whole protocol that is very similar to that of $\#SAT$ we here address the two problems of Exercise 2 and explain how to modify the protocol accordingly.

Small problem: The small problem is that the arithmethization of $\Psi = \forall x_1, \exists x_2, \ldots, Q_n x_n \varphi(x_1, \ldots, x_n)$ which equals

$$\prod_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \prod_{b_3 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} P_\varphi(b_1, \ldots, b_n) \tag{4}$$

can take values which is doubly exponential $O(2^{2^n})$. Hence the prover cannot just send K so that we need to verify that $K = (4)$. Indeed, sending K can require exponentially many bits and the verifier can only read polynomially many bits.

The fix is as follows. Instead of only sending K, the prover sends a prime $p : 1 \leq p \leq 2^{poly(n)}$ and $K : 1 \leq K \leq p$ and the task is to design an interactive proof that verifies $(4) = K$ modulo p. We note that such a prime p (and K) must exist because if every prime $\leq 2^{poly(n)}$ divides (4) then (4) would be greater than $O(2^{2^n})$ which is a contradiction (this follows from the prime number theorem).

Slightly bigger problem: The degree of the polynomial $s(X_1)$ that the prover sends and is supposed to equal $\sum_{b_2 \in \{0,1\}} \prod_{b_3 \in \{0,1\}} \cdots \sum_{b_n \in \{0,1\}} P_\varphi(X_1, \ldots, b_n)$ can be exponential as multiplications increase the degree. Again, the prover cannot communicate the coefficients of this polynomial by using polynomially many bits.

The fix is to rewrite the formula Ψ into an equivalent formula so that its corresponding polynomial is not of too large degree in each step. The rewriting is as follows:

- The boolean formula Ψ is re-written from right to left by considering each quantifier one at a time and adding more quantifiers and variables in the process. (Note that only quantifiers present in the original formula are considered in the process and not the newly added ones).
- If the quantifier being considered is $\exists$ then do nothing (sums are not harmful for the degree).
- Otherwise, represent the formula being considered as $\varphi(x_1, x_2, \ldots, x_{i-1}) = \forall x_i (x_1, x_2, \ldots, x_i)$. Now rewrite the formula as

  $$\varphi'(x_1, \ldots x_{i-1}) = \forall x_i \exists x_1^i, \ldots, x_i^i : \left(\wedge_{j=1}^i x_j = x_j^i\right) \wedge \varphi(x_1^i, \ldots, x_i^i).$$

- It is easy to see that both formulaes are equivalent and that the arithmetization. Furthermore, the new formula satisfies the following property: let $y_1, \ldots, y_N$ be the variables (of the new formula) sorted in the order of appearance, then for every variable y_i there is at most a single universal quantifier ($\forall$) involving y_j (for $j > i$).

  Note that, as pointed out by Jonathan, the arithmetization of this formula may still not be of polynomial degree. However, one can prove that running the subset sum protocol on such a formula as we did for #SAT with the modification that we check $s(0) \cdot s(1) = K$ for product operations results in that the prover only sends polynomials of polynomial degree. (You are asked to prove this in Homework III.)

Example 1 *We give an example of the rewriting of the formula:*

$$\forall x_1 \exists x_2 \forall x_3 \exists x_4 \varphi(x_1, x_2, x_3, x_4)$$

becomes

$$\forall x_1 \exists x_2 \forall x_3 (\exists x_1^3, x_2^3, x_3^3 : (x_1^3 = x_1 \wedge x_2^3 = x_2 \wedge x_3^3 = x_3) \exists x_4 \varphi(x_1, x_2^3, x_3^3, x_4^3)$$

when "rewriting the $\forall x_3$ *quantifier" and finally it becomes*

$$\forall x_1 \exists x_1^1 : (x_1^1 = x_1) \exists x_2 \forall x_3 (\exists x_1^3, x_2^3, x_3^3 : (x_1^3 = x_1^1 \wedge x_2^3 = x_2 \wedge x_3^3 = x_3) \exists x_4 \varphi(x_1^1, x_2^3, x_3^3, x_4^3)$$

when rewriting the $\forall x_1$ *quantifier. Note that the introduction of "fresh" variables leads to an arithmetization of low (polynomial) degree.*