6.841 Advanced Complexity Theory April 4, 2005

Lecture 16

Lecturer: Madhu Sudan *Scribe: Kyomin Jung*

1 Overview

In this lecture, we will show that

- $IP \subseteq PSPACE$
- $\#P \subseteq IP$
- towards $PSPACE \subseteq$ IP.

Remind that IP is the class of languages L such that $L \in IP$ if and only if there exists a probabilistic polynomial time verifier V such that

- $w \in L \rightarrow \;\; \exists$ Prover P s.t. the probability that V accepts w in interaction with P is $\geq \frac{3}{4}$
- $x \notin L \rightarrow \;\; \forall$ Prover P, the probability that V accepts w in interaction with P is $\leq \frac{1}{4}$

Note that there is no computational limitation on P. In fact, above two-sided-error definition can be replaced by one-sided-error. And the constants $\frac{3}{4}$ and $\frac{1}{4}$ can be replaced by $1-(\frac{1}{2})^{poly(n)}$ and $(\frac{1}{2})^{poly(n)}$.

2 IP in PSPACE

In this section we show that $IP \in PSPACE$. To this end, think of a language $L \in IP$. We want to show that we can check whether a given string w is in L or not using polynomial space. Let V be the corresponding verifier of L, and w be a string we want to check. Then, we will show that we can compute whether there exists a prover P such that

$$\Pr_R[verdict(P \leftrightarrow V) = 1] \geq \frac{3}{4}$$

in polynomial space. For a given V and w, we will compute the maximum value of $\Pr_R[verdict(P \leftrightarrow V) = 1]$ that can be achieved from some P. If the probability is higher than $\frac{3}{4}$, then we see that $w \in L$. Otherwise we see that $w \notin L$. Now we explain how we obtain the maximum probability an optimal prover can achieve. Let $P(w, q_1, a_1, \ldots, q_i, a_i)$ be the maximum value of probability that w is accepted under the specified message history up to i th stage, where maximum value is taken over all the possible P that is going to be decided after the ith stage. We can define $P(w, q_1, a_1, \ldots, q_i)$ in the same way. Then we can compute these probabilities recursively from $P(w, q_1, a_1, \ldots, q_{k(n)}, a_{k(n)})$'s to $P(w)$, where $k(n)$ is the number of stages in IP. Clearly, we know the values $P(w, q_1, a_1, \ldots, q_{k(n)}, a_{k(n)})$, which will be 0 or 1 according to the value of $verdict()$. Given all the values of the form $P(w, q_1, a_1, \ldots, q_i, a_i)$, we define

$$P(w, q_1, a_1, \ldots, q_i) = \max_{a_i} P(w, q_1, a_1, \ldots, q_i, a_i).$$

And given all the values of the form $P(w, q_1, a_1, \ldots, q_i)$, we define

$$P(w, q_1, a_1, \ldots, q_{i-1}, a_{i-1}) = \Pr_{R_i} P(w, q_1, a_1, \ldots, q_i).$$

Note that q_i depends on R_i, and the probability is taken over R_i. By this way we obtain $P(w)$, and it is exactly the maximum value of $\Pr_R[verdict(P \leftrightarrow V) = 1]$ over all possible P. Note that here, there is no computational restriction on P. Now the depth of recursion is proportional to the number of stages in IP that is polynomial over n. So we can compute $P(w)$ in polynomial space, and we obtain the required result.

3 #P in IP

In this section, we prove that $\#P \in IP$. Given a cnf formula ϕ, we want to count the number of satisfying assignment of ϕ, by designing an interactive protocol. Let n be the number of boolean variables in ϕ, and m be the number of clauses in ϕ. Let $\#(\phi)$ be the number of satisfying assignment of ϕ. Similarly, let $\#(\phi|a_1, a_2, \ldots, a_j)$ be the number of satisfying assignment of ϕ with the condition that $x_1 = a_1, x_2 = a_2, \ldots, x_j = a_j$. Then,

$$\#(\phi) = \#(\phi|0) + \#(\phi|1) = \#(\phi|0,0) + \#(\phi|1,0) + \#(\phi|0,1) + \#(\phi|1,1) = ...$$

, And so on. We obtain these relations until all the boolean variables are assigned. A simple way to design an interactive protocol is, verifier asks the prover all the values of $\#(\phi|\ldots)$ from the top to the bottoms, and the verifier checks whether all the above relations hold, and when all the variables are assigned, check that the prover's answer is the actually the satisfiability value (0 or 1) of ϕ with that assignment. But the problem is, this protocol requires exponentially many calculations for the verifier. So, instead of checking all the branches of the tree, we will check "randomly" selected one branch of it. Note that in the real tree, there are only two branches for each node. But, instead, by thinking $\#(\phi|_{...})$ as a function in a field Z_p for some large p which we will specify later, we can select a random branch from possible p branches.

Arithmetization of Logical expressions

$$\begin{array}{rcl}
\textbf{Logical expression} & \rightarrow & \textbf{In field } Z_p \\
\text{True/False} & \rightarrow & 1/0 \\
\text{variables } x_i & \rightarrow & x_i \\
\text{literals } \neg x_1 & \rightarrow & 1 - x_1 \\
\text{clauses } C_i = (x \vee y \vee \neg z) & \rightarrow & P_i = 1 - (1-x)(1-y)z \\
\text{SAT } C_1 \wedge C_2 \ldots C_m & \rightarrow & P = \prod_{i=1}^{m} P_i
\end{array} \tag{1}$$

Then P is a degree $\leq 3m$ polynomial over n variables. Choose prime $p \in_R [10nm, 20nm]$. From now on, all the calculations are carried out in the field Z_p.

Then, we obtain

$$\begin{array}{rcl}
\phi(x_1, \ldots, x_n) & = & P(x_1, \ldots, x_n), \\
\#(\phi|a_1, a_2 \ldots, a_j) & = & \displaystyle\sum_{(x_{j+1}, \ldots x_n) \in \{0,1\}^{n-j}} P(a_1, \ldots, a_j, x_{j+1}, \ldots, x_n).
\end{array} \tag{2}$$

Note that, for P, x_i's does not need to be just 0 or 1. we can calculate P for any $x_i \in Z_p$.

Let

$$f_1(x_1) \triangleq \sum_{(x_2, \ldots x_n) \in \{0,1\}^{n-1}} P(x_1, x_2, \ldots, x_n).$$

Then, $f_1(x_1)$ is a polynomial in x_1 with degree $\leq 3m$. Similarly, we will define $f_i(x_i)$ in the protocol defined below. Now the interactive protocol is as follows:

The protocol for #SAT

- The verifier asks to the prover $\#(\phi)$, $\#(\phi|0)$, and $\#(\phi|1)$.

- The prover responds with Q, Q_0 and Q_1.
- The verifier verifies that $Q \leq 2^n$ and $Q = Q_0 + Q_1$. If any of them does not hold, it rejects. The verifier asks for the polynomial $f_1(x_1)$.
- The prover responds with $h_1(x)$ by sending the coefficients.
- The verifier verifies that $h_1(x)$ is of degree $\leq 3m$, $Q_0 = h_1(0)$, $Q_1 = h_1(1)$. if any of them does not hold, it rejects. Now the verifier picks a random number $a_1 \in Z_p$ and sends it to the prover, asking it to prove that the value $h_1(a_1)$ would evaluate to $f_1(a_1)$.

  Recursively,
- (in stage $2i+1$) The verifier is trying to verify that $h_{i-1}(a_{i-1}) = \sum_{s \in \{0,1\}^{n-i+1}} P(\mathbf{a}, s)$, where the vector $\mathbf{a} = a_1 a_2 \ldots a_{i-1}$ represents the sequence of random choices that the verifier made from the first to the current stage. Now the verifier asks for the polynomial $f_i(x_i) \triangleq \sum_{s' \in \{0,1\}^{n-i}} P(\mathbf{a}, x_i, s')$.
- (in stage $2i+2$) The prover responds with a $h_i(x)$ by giving the coefficients.
- (in stage $2i+3$) The verifier verifies that $h_i(x)$ is of degree $\leq 3m$ and $h_{i-1}(a_{i-1}) = h_i(0) + h_i(1)$; if any of them does not holds, it rejects. Now pick a random $a_i \in Z_p$ and sends it to the prover, asking it to prove that the equation $h_i(a_i) = f_i(a_i)$ holds. (asking for the polynomial $f_{i+1}(x_{i+1})$)
- At the end, the verifier checks whether $h_n(a_n) = \phi(a_1, a_2, \ldots, a_n)$. If it holds, accept. If not, reject.

Note that the verifier is doing his calculation in polynomial time.

Completeness

Trivial. If Prover responds with correct answer, the verifier accepts probability 1.

Soundness

Suppose that a crooked prover tries to prove that the number of satisfying assignments is Q, that is different from $\#(\phi)$. Then at the phase 1, the prover replies with Q, Q_0, Q_0. We may assume that $Q_0 + Q_1 = Q$ and $h_1(0) = Q_0$ and $h_1(1) = Q_1$ because, otherwise it will be rejected. Note that then, $h_1(x) \neq f_1(x)$ because $h_1(0) \neq f_1(0)$ or $h_1(1) \neq f_1(1)$. Now, then by choosing $a_1 \in Z_p$ at random,

$$\Pr[h_1(a_1) = f_1(a_1)] < \frac{3m}{10nm},$$

because a non-zero polynomial of degree at most $3m$ has at most $3m$ zeros in Z_p. Now inductively, under assumption that $h_i(a_i) \neq f_i(a_i)$, same argument is used to show that

$$\Pr[h_{i+1}(a_{i+1}) = f_{i+1}(a_{i+1}) | h_i(a_i) \neq f_i(a_i)] < \frac{3}{10n}.$$

So,

$$\Pr[h_n(a_n) \neq f_n(a_n)] > (1 - \frac{3}{10n})^n \approx \frac{7}{10}$$

Note that $f_n(a_n) = \phi(a_1, a_2, \ldots a_n)$. So we obtain the soundness $\frac{3}{10}$.

4 Toward PSPACE in IP

Note that in the proof of $\#P \in IP$, what we used is the self reducibility of the SAT formula in obtaining the number of satisfying assignments. So we can apply similar argument to PQBF, which also has the self reducibility property, to show that $PSPACE \in IP$.

Suppose that we are given a quantified boolean formula $\psi = \exists x_1 \forall x_2 \ldots Q_n x_n \phi(x_1, x_2 \ldots, x_n)$.

We need to find a rule to construct polynomials $(Q_0, Q_1, \ldots, Q_n)$ such that,

- $Q_0(x_1, \ldots, x_n)$ are easily computable.
- Q_i equals sum/product of Q_{i-1} at two points.
- $\deg(Q_0, Q_1, \ldots, Q_l) \leq D = \text{poly}(n)$

Then we can interactively prove that given ψ is true or not using similar protocol used in the previous section. We will discuss more about this in the next lecture.