COM S 6810 Theory of Computing March 3, 2009

Lecture 13: Unique SAT, Interactive Proofs

Instructor: Rafael Pass *Scribe: Hu Fu*

1 Unique SAT

Before talking about the main theorem today, we first mention an important result that we regretfully shall skip:

Theorem 1 *(Toda)* $PH \in \mathcal{P}^{\#\mathcal{P}}$.

Intuitively, this result shows that the ability to solve $\#\mathcal{P}$ problems exactly is powerful, which is in some sense counter to what we showed last time, i.e., approximate counting in $\#\mathcal{P}$ can be done randomly with an oracle to $\mathcal{NP}$.

Now we introduce the main question today: Is it easier to find a satisfying assignment for a boolean formula if we know that the solution is unique?

The question can be equivalently formed in the fashion of a decision problem: Is it easier to decide whether a boolean formula is satisfiable if we know that it can have at most one satisfying assignment?

We introduce a few notations: Given a boolean formula φ, we write $\#\varphi$ to denote the number of satisfying assignments to φ. Let $UniqueSAT$ be the language of all boolean formulas that have exactly one satisfying assignment, i.e, $\{\varphi : \#\varphi = 1\}$. Let $fewSAT$ be the language of boolean formulas that have at most 100 satisfying assignments if there is any, i.e., $\{\varphi : 0 \leq \#\varphi \leq 100\}$.

Here is the main theorem:

Theorem 2 (Valiant-Vazirani) *: If* $UniqueSAT \in \mathcal{P}$*, then* $SAT \in \mathcal{RP}$.

Equivalently, and more concretely, the result can be restated as follows:

There exists a polynomial p and a probabilistic polynomial time Turing machine A such that for any boolean formula φ, if $\varphi \in SAT$, then

$$Pr[\varphi \leftarrow A(\varphi) : \varphi' \in UniqueSAT] \geq \frac{1}{p(|\varphi|)},$$

where $\varphi' \leftarrow A(\varphi)$ means that A outputs φ' on the input φ; and if $\varphi \notin SAT$, then

$$Pr[\varphi' \leftarrow A(\varphi) : \varphi' \in SAT] = 0.$$

Note that the probability $\frac{1}{p(|\varphi|)}$ is enough to guarantee the reduced problem to be in $\mathcal{RP}$, by repeatedly running A on φ for a polynomial number of times.

Before proving the main theorem, let's first consider a simpler problem: given $\varphi \in fewSAT$, can we design a probabilistic polynomial time Turing machine B that reduces φ to φ' such that

$$\#\varphi > 0 \quad \Rightarrow \quad Pr[\varphi' \leftarrow B(\varphi) : \varphi' \in UniqueSAT] \geq \frac{1}{100},$$
$$\#\varphi = 0 \quad \Rightarrow \quad Pr[\varphi' \leftarrow B(\varphi) : \varphi' \in UniqueSAT] = 0?$$

After a moment's thought, we see that this can be easily done. We can first guess $\#\varphi$: assume that φ has m solutions, then output φ_m defined as

$$\varphi_m = \bigwedge_{i=1}^{m} \varphi(\mathbf{x}_i) \wedge (\mathbf{x}_1 < \mathbf{x}_2 < \cdots < \mathbf{x}_m),$$

where "<" is the natural lexical ordering on the assignments. The constraint $\mathbf{x}_1 < \mathbf{x}_2 < \cdots < \mathbf{x}_m$ forces all assignments to be different, and if φ does have m solutions, then φ_m has exactly one solution. This means that, whenever we guessed m correctly, the output is guaranteed to be in $UniqueSAT$ (when $m > 0$); and when $\#\varphi = 0$, however we guessed m, $\#\varphi_m = 0$. But since φ is in $fewSAT$, there are only 100 choices for m, and hence the probability that we guessed m correctly is at least $\frac{1}{100}$ when $\#\varphi > 0$. In short, the probabilistic polynomial time Turing machine B we want is this: Given φ, guess an m from $\{1, 2, \cdots, 100\}$, then output φ_m.

Therefore the problem boils down to this: given a boolean formula φ, can we randomly reduce it to a formula in $fewSAT$ with a decent probability of correctness? Since there can be at most 2^n solutions (n is the number of variables involved in φ), and the set $\{1, 2, \cdots, 2^n\}$ can be divided into n intervals:

$$\{2^0\}, \{2^1, 2^2 - 1\}, \{2^2, 2^2 + 1, \cdots, 2^3 - 1\}, \cdots, \{2^{n-1}, 2^{n-1} + 1, \cdots, 2^n\},$$

if we can know which interval $\#\varphi$ lies in, then we will likely be able to use the hashing technique introduced last time to reduce the number of satisfying assignments. This will be the main idea of our proof.

Now we define the probabilistic polynomial time Turing machine A using B as a subroutine: Given a boolean formula φ with n variables; A chooses i uniformly randomly from $\{1, 2, \cdots, n\}$; if $i \leq 3$, then output $B(\varphi)$; otherwise, A picks h randomly from $H_{n,i-3}$ where $H_{n,i-3}$ is the set of hashing functions we introduced last lecture that map $\{0,1\}^n$ to $\{0,1\}^{i-3}$; let $\varphi'(x) = \varphi(x) \wedge h(x) = 0^{i-3}$, then outputs $B(\varphi')$.

Let us analyze the performance of A. If $\#\varphi = 0$, then all the operations that A (and B) performs does not introduce satisfying assignments, and hence $Pr(\varphi' \leftarrow A(\varphi) : \varphi' \in UniqueSAT) = 0$.

How about when $\#\varphi > 0$? As described, there are n intervals, and if we randomly guess $i \in [n]$, the probability that $2^i \leq \#\varphi \leq 2^{i+1}$ is at least $\frac{1}{n}$. Hence from now on assume that this relation holds. If $i \leq 3$, then

$$Pr(\varphi' \leftarrow B(\varphi) : \varphi' \in UniqueSAT) \geq \frac{1}{100}.$$

If $i > 3$, A applies a hashing function on φ. Recall the theorem we proved about these hashing functions:

Theorem 3 *(Hashing) Let $\epsilon > 0$, $S \subseteq \{0,1\}^n$ such that $|S| \geq \frac{2^m}{\epsilon^3}$, then*

$$\underset{h \in H_{n,m}}{Pr}\left[(1-\epsilon)\frac{|S|}{2^m} \leq |\{x \in S | H(x) = 0\}| \leq (1+\epsilon)\frac{|S|}{2^m}\right] > 1 - \epsilon.$$

Here we have $m = i - 3$, $|S| = \#\varphi$, and $|\{x \in S\}|h(x) = 0| = \#\varphi'$. The requirement that $|S| \geq \frac{2^m}{\epsilon^3}$ becomes $\#\varphi \leq \frac{2^{i-3}}{\epsilon^3}$; but as we have assumed that $2^i \leq \#\varphi \leq 2^{i+1}$, it is enough to have $\epsilon \geq \frac{1}{2}$. We can fix $\epsilon = \frac{1}{2}$. The conclusion of the theorem becomes

$$Pr[\varphi' \leftarrow A(\varphi) : 4 \leq \#\varphi' \leq 24] \geq \frac{1}{2}.$$

This says that with at least half of the probability, the output φ' is in $fewSAT$. Then with probability $\frac{1}{100}$, $B(\varphi')$ is in $UniqueSAT$.

To sum up, when $\#\varphi > 0$, with probability at least $\frac{1}{n} \cdot \frac{1}{2} \cdot \frac{1}{100} = \frac{1}{200n}$, $A(\varphi)$ is in $UniqueSAT$. This completes the proof for the theorem.

2 Interactive Proof

Before starting the new topic, let us look at the definition of $\mathcal{NP}$: $\mathcal{NP}$ is the class of languages L such that for each L there exists a polynomial-time verifier V such that

$$x \in L \quad \Rightarrow \quad \exists \pi, V(x,\pi) = 1,$$

$$x \notin L \quad \Rightarrow \forall \pi, V(x,\pi) = 0.$$

A natural relaxation of this definition gives another class of languages we call $\mathcal{MA}$: $\mathcal{MA}$ is defined as the class of languages L such that for each L there exists a probabilistic polynomial time Turing machine V such that

$$x \in L \quad \Rightarrow \quad \exists \pi, Pr[V(x,\pi) = 1] \geq \frac{2}{3},$$

$$x \notin L \quad \Rightarrow \quad \forall \pi, Pr[V(x,\pi)=1] \leq \frac{1}{3}.$$

The first requirement above is called completeness, and the second soundness.

Looking at these definitions, we notice that the proofs are all "written". Can we have proof systems without necessarily writing down the proofs? This motivates the following definition of interactive proofs:

Definition 1 (P,V) *is called an interactive proof for a language* L *if* P *is an interactive Turing machine (ITM), and* V *is a probabilistic polynomial-time ITM, and*

$$\bullet\textit{[Completeness]} \quad x \in L \quad \Rightarrow \quad Pr[(P,V)(x)=1] \geq \frac{2}{3}.$$

$$\bullet\textit{[Soundness]} \quad x \notin L \quad \Rightarrow \quad \forall ITM\ P^*, Pr[(P^*,V)(x)=1] \leq \frac{1}{3}.$$

We will give a precise definition for "interactive Turing machine" in the next lecture. For now, notice that in the soundness requirement of this definition, it is not adequate to just state that $Pr[(P,V)(x)=1] \leq \frac{1}{3}$. Moreover, this definition is a generalization of $\mathcal{NP}$, since the ITM P can send to V the proof for an instance x, and then V can check it and outputs the correct answer with probability 1.