6.841/18.405J: Advanced Complexity Theory	March 10, 2002
Lecture 10	
Lecturer: Madhu Sudan	*Scribe: David Pritchard*

The topics that will be covered today are:

1. Completing the proof of the Valiant-Vazirani Theorem
2. Introduction to the "Counting Problem" class $\#\mathsf{P}$
3. Begin the proof of Toda's Theorem, $\mathsf{PH} \subseteq \mathsf{P}^{\#\mathsf{P}}$

1 SAT and USAT

Last class we introduced the problem of unique satisfiability, USAT. Unlike the other problems we have so far discussed, USAT is not a language but instead a "promise problem": we want a TM that will decide if a given formula ϕ is satisfiable or not *under the condition that we promise the formula is either unsatisfiable or uniquely satisfiable* (that is, $\{x|\phi(x)$ is true$\}$ is $\emptyset$ or has exactly one element). It is evident that USAT is, in a sense, easier than SAT since anything that decides SAT will also decide USAT. However, if USAT is easy enough then another surprising conclusion is true:

Theorem 1 (Valiant-Vazirani) *USAT $\in$ P implies $\mathsf{NP} = \mathsf{RP}$.*

We prove this by means of the following lemma, which effectively states that there is a randomized reduction from SAT to USAT:

Lemma 2 *In polynomial time we can probabilistically reduce a formula $\phi \in SAT$ to another formula ψ such that $\phi \notin SAT \Rightarrow \psi \notin SAT$, and $\phi \in SAT \Rightarrow$ with probability $1/\text{poly}(n), \psi$ is uniquely satisfiable.*

(Here n in the number of variables in *phi*). Lemma 2 implies Theorem 1 because, if USAT $\in \mathsf{P}$, then our RP-machine for solving SAT could just perform this reduction polynomially many times (as many as is needed to amplify the probability of finding at least one correct reduction to a constant) and accept iff our USAT solver accepts any of these reductions[1]. We proceed, then, to prove the Lemma, as outlined in the previous lecture.

1.1 Proving SAT reduces to USAT

Proof Strategy: Let $\psi(x) = \phi(x) \wedge h(x) = 0$ for a suitably chosen h.

First of all, we want to pick h from a pairwise independent, nice (as defined in the last lecture) family of functions: so we will pick our h from the family $\{h_{A,b} : x \mapsto Ax + b\}$ where all operations are done in $\mathbb{Z}_2$, A is an $m \times n$ matrix, b is an m-element vector, and all elements of A and b are chosen uniformly from $\{0, 1\}$. (Note that these functions take elements of $\mathbb{Z}_2^n$ to elements of $\mathbb{Z}_2^m$, so the phrase "$h(x) = 0$" is a vector equality.) Note that if ϕ is unsatisfiable, it is immediate that ϕ is unsatisfiable also: so henceforth, we will concern ourselves only with the case that *phi* has one or more satisfying assignments.

The question remains, what should m be? Denote the set of all satisfying assignments of ϕ by S, and let $M = |S|$. If we know the value of M, it turns out (as we will see shortly) that taking m such that $2^{m-2} \leq M \leq 2^{m-1}$ is a good choice. However, there is no cheap way to even approximate M, so what we do is the following:
Choice of m: Choose m randomly (and uniformly) from $\{2, 3, \ldots, n+1\}$.

Since ϕ has between 0 and 2^n satisfying assignments (its n boolean variables can only take on 2^n distinct values), we have a $1/n$ chance of picking the correct m.

[1]Since ψ may be multiply-satisfiable it is possible that we are giving the USAT solver problems outside of $\text{USAT}_{\text{NO}} \bigcup \text{USAT}_{\text{YES}}$; however the one-way error of our reduction means that it could only possibly accept formulae reduced from a satisfiable ϕ.

1.2 How Well Does This Work?

In the $1/n$ chance that we have picked the correct m, we would like to know how likely it is that our new formula has only one satisfying assignment.

Lemma 3 *If $2^{m-2} \leq M \leq 2^{m-1}$, then ψ is uniquely satisfiable with probability $\geq 1/8$.*

Proof

For a given boolean n-vector x in S, Let G_x be the event $(h(x) = 0 \wedge \forall y \in S \backslash \{x\}, h(y) \neq 0)$. Note that the G_x are mutually exclusive. Now, call $y \in S$ "bad for x" if $h(x) = h(y) = 0$.

We see that Pr[y is bad for x] is $1/2^m \cdot 1/2^m$, since Pr[x=0]=$1/2^m$, Pr[y=0]=$1/2^m$, and h was selected from a pairwise independent family of hash functions. Thus, by using the union-bound, Pr[$\exists y$ such that y is bad for x] $\leq |S - \{x\}|/2^{2m} < M/2^{2m}$. Hence

$$\begin{aligned}
\Pr[G_x] &= \Pr[h(x) = 0 \wedge \forall y \in S\backslash\{x\}, h(y) \neq 0] && (1)\\
&= \Pr[h(x) = 0 \wedge \neg(\exists y : y \text{ is bad for } x)] && (2)\\
&= \Pr[h(x) = 0] - \Pr[\exists y : y \text{ is bad for } x] && (3)\\
&> \frac{1}{2^m} - \frac{M}{2^{2m}} && (4)\\
&= \frac{(2^m - M)}{2^{2m}} && (5)
\end{aligned}$$

Which in turn gives us

$$\begin{aligned}
\Pr[\psi \text{ is uniquely satisfiable}] &= \Pr[\exists x : G_x] && (6)\\
&> \frac{M \cdot (2^m - M)}{2^{2m}} && (7)\\
&= \frac{M}{2^m} \cdot \frac{2^m - M}{2^m} && (8)\\
&\geq \frac{2^{m-2}}{2^m} \cdot \frac{2^{m-1}}{2^m} && (9)\\
&= 1/8 && (10)
\end{aligned}$$

(where step 9 comes from the inequality $2^{m-2} \leq M \leq 2^{m-1}$), which completes the proof. ■

Now, to complete the proof of the lemma, we need only observe that this implies Pr[ψ is uniquely satisfiable] $\geq$ Pr[m was chosen correctly]$\cdot$Pr[ψ is uniquely satisfiable $|m$ is chosen correctly] $> 1/8n$.

1.3 Finishing Up

We still need to show that $h(x)$ can actually be expressed as a boolean formula. By using a process similar to that described in the proof of the Cook-Levin theorem, we can rewrite $h(x)$ as $\exists y : \eta(x, y)$. Roughly speaking, $\eta(x, y)$ means "y represents a computation of a TM calculating $h(x)$, and $h(x) = 0$". Furthermore it is evident from this process that at most one such y exists, so our new, uniquely satisfiable (or unsatisfiable) formula would be $\psi(x, y) = h(x) \wedge \eta(x, y)$.

1.4 Remarks

This reduction is pretty remarkable, but there are stronger related reductions that remain open questions:

- Is there a non-probabilistic reduction from SAT to USAT?

- Is there a high-probability reduction from SAT to USAT?
- Is there a reduction from SAT to USAT whose accuracy we can check in polynomial time?

All of these questions have applications in cryptography, since USAT can be used to define a certain class of one-way functions (as mentioned in the last lecture).

2 The Classes $\#\mathsf{P}$ and $\mathsf{P}^{\#\mathsf{P}}$

Consider the problem of counting how many satisfying assignments a particular boolean formula has. How difficult is this problem? How can we relate it to other problems? These questions motivate the definition of the class $\#\mathsf{P}$:

Definition 4 *$\#P$ is the collection of all functions $f : \{0,1\}^* \to \mathbb{Z}_{\geq 0}$ defined by $f(x) = |\{y|M(x,y) \text{ halts}\}|$, where M is any polynomial-time (in terms of the first argument) TM.*

An equivalent definition would be to express $f(x)$ as the number of accepting paths for a poly-time NTM on input x. Note that this is a class of functions, not of languages; but we can change a function f into the language $L_f = \{< x, f(x) > | x \in \mathbb{Z}\}$ and a language into a function to $\{0,1\}$, so we mix the two freely.

The class $\mathsf{P}^{\#\mathsf{P}}$ is the class of polynomial-time computable functions on TMs that have oracle access to all $\#\mathsf{P}$ functions (its queries are of the form $< M, x >$ where x is any string and M is a machine of the type described above).

The reductions of Cook and Karp are useful in proving completeness in this class, since they can be made preserve the number of second-arguments for any particular input; because of this, some of the complete functions for $\mathsf{P}^{\#\mathsf{P}}$ are:

- #SAT: How many satisfying assignments does a formula ϕ have?
- #HAMCYC: How many hamiltonian cycles does a directed graph have?
- #CYC: How many (simple?) cycles does a directed graph have? (Can be obtained from #HAMCYC).
- How many matchings does a given bipartite graph have?
- What is the *permanent* of a given matrix? (See below for definition).

The fact that #CYC is complete for this class is quite surprising, since determining whether a graph on n nodes has a cycle can be done in $O(n^4)$ time. The *permanent* of an n-by-n matrix $A = \{a_{i,j}\}$ is given by the formula

$$\mathrm{perm}A = \sum_{\pi \in S_n} \prod_{i=1}^{n} a_{i,\pi(i)}$$

where $\pi \in S_n$ means that we take the sum over all permutations π of $\{1, 2, \ldots, n\}$. It is used in physics, for example to compute the energies of certain systems. The determinant can be defined by an almost-identical formula:

$$\det A = \sum_{\pi \in S_n} (-1)^{l(\pi)} \prod_{i=1}^{n} a_{i,\pi(i)}$$

where $l(\pi)$ is the "length", or number of inversions in π. However, despite the formulas' similarities, the permanent is $\mathsf{P}^{\#\mathsf{P}}$-complete and the determinant can be computed in $O(n^3)$ time (by using Gaussian elimination, or LU-decomposition)!

We can see the upper bound $\mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{PSPACE}$ by a straightforward simulation argument; whether this inequality is an equality is an open question. It is clear that $\mathsf{NP}, \mathsf{coNP} \subseteq \mathsf{P}^{\#\mathsf{P}}$ since we can just query the oracle with a SAT TM and return yes if at least one (respectively, all) of the computation paths accept. $\mathsf{BPP}, \mathsf{RP}, \text{co-}\mathsf{RP} \subseteq \mathsf{P}^{\#\mathsf{P}}$ since we can just compute how many of the possible branches accept and take the majority response. What about Σ_2^P? Toda's theorem answers this question:

Theorem 5 (Toda) $\forall i, \Sigma_i^P \subseteq P^{\#P}$.

We will (start to) prove this theorem after we introduce some notation.

3 Complexity Class Operators and Toda's Theorem

An "operator" can be thought of a higher-order function: its input and output are complexity classes (sets of languages). We write the result of applying operator $\mathcal{O}$ to the class $\mathcal{C}$ as $\mathcal{O} \cdot \mathcal{C}$. A particularly simple operator is $\neg$, defined by $\neg \cdot \mathcal{C} = \{\overline{L} | L \in \mathcal{C}\}$. In words, $\neg \cdot \mathcal{C}$ is the class of complements of languages in $\mathcal{C}$. Several other operators include:

- $\exists : \mathcal{C} \mapsto \{\{x | \exists y, (x, y) \in L\} | L \in \mathcal{C}\}$
- $\forall : \mathcal{C} \mapsto \{\{x | \forall y, (x, y) \in L\} | L \in \mathcal{C}\}$
- $\bigoplus : \mathcal{C} \mapsto \{\{x | \text{ for an odd number of } y, (x, y) \in L\} | L \in \mathcal{C}\}$
- $\mathsf{BP} : \mathcal{C} \mapsto \{L' | \exists L \in \mathcal{C} : (x \in L' \to L \text{ contains at least a fraction } c(n) \text{ of } (x, y), x \notin L' \to L \text{ contains at most a fraction } s(n) \text{ of } (x, y), c(n) - s(n) \geq 1/\text{poly}(n)\}$

Although these definitions are quite technical, they are most easily understood by a few examples:

- $\exists \cdot \mathsf{P} = \mathsf{NP}$
- $\forall \cdot \mathsf{P} = \mathsf{coNP}$
- $\mathsf{BP} \cdot \mathsf{P} = \mathsf{BPP}$
- $\neg \cdot \mathsf{NP} = \mathsf{coNP}$
- $\exists \cdot \Sigma_3^P = \Sigma_3^P$
- $\forall \cdot \Sigma_3^P = \Pi_4^P$

3.1 Overview of Toda's Theorem

In order to show that all $\Sigma_i^P \subseteq \mathsf{P}^{\#\mathsf{P}}$, we proceed in several steps:

1. $\Sigma_i^P \subseteq \mathsf{BP} \cdot \bigoplus \cdot \Pi_{i-1}^P$, and $\Pi_i^P \subseteq \mathsf{BP} \cdot \bigoplus \cdot \Pi_{i-1}^P$ (Extends Valiant-Vazirani.)
2. $\mathsf{BP} \cdot \bigoplus \cdot \mathsf{P}$ amplifies error (Subtle.)
3. $\bigoplus \cdot \mathsf{BP} \cdot \bigoplus \cdot \mathsf{P} \subseteq \mathsf{BP} \cdot \bigoplus \cdot \bigoplus \cdot \mathsf{P} \subseteq \mathsf{BP} \cdot \bigoplus \cdot \mathsf{P}$ (Surprising, but straightforward.)
4. $\mathsf{BP} \cdot \mathsf{BP} \cdot \bigoplus \cdot \mathsf{P} \subseteq \mathsf{BP} \cdot \bigoplus \cdot \mathsf{P}$ (Not surprising, straightforward.)
5. $\mathsf{BP} \cdot \bigoplus \cdot \mathsf{P} \subseteq \mathsf{P}^{\#\mathsf{P}}$ (Completely separate theorem.)

Once we have shown all of this, an easy induction proof shows that each of the classes Σ_i^P can be collapsed into $\mathsf{BP} \cdot \bigoplus \cdot \mathsf{P}$, which means that they are all subseteqs of $\mathsf{P}^{\#\mathsf{P}}$.

3.2 Proof of Step 1

We need to prove: $\Sigma_i^P \subseteq \mathsf{BP} \cdot \bigoplus \cdot \Pi_{i-1}^P$

We will show that i-TQBF is in $\mathsf{BP} \cdot \bigoplus \cdot \Pi_{i-1}^P$, which is sufficient since this is a complete problem for the class. Consider the formula $\exists x_1 \forall x_2 \ldots \mathsf{Q}_i x_i \phi(x_1, x_2, \ldots x_n)$. We pick a pairwise-independent, nice hash function h and consider the number of solutions to $\forall x_2 \ldots \mathsf{Q}_i x_i \phi(x_1, x_2, \ldots x_n) \wedge h(x_1) = 0$. With inverse polynomial probability, there will be exactly zero or one solutions: so applying $\bigoplus$ to this problem gives enough information to solve the original i-TQBF problem, and we are done.

We also need to prove: $\Pi_i^P \subseteq \mathsf{BP} \cdot \bigoplus \cdot \Pi_{i-1}^P$. Consider

$$\begin{aligned}
\Pi_i^P &= \neg \cdot \Sigma_i^P && (11)\\
&\subseteq \neg \cdot \mathsf{BP} \cdot \bigoplus \cdot \Pi_{k-1}^P && (12)\\
&= \mathsf{BP} \cdot \neg \cdot \bigoplus \cdot \Pi_{k-1}^P && (13)\\
&= \mathsf{BP} \cdot \bigoplus \cdot \Pi_{k-1}^P && (14)
\end{aligned}$$

We just proved the first assertion; the second assertion follows, roughly, from the fact that BP is symmetric in allowing errors on both sides (ie both falsely accepting and falsely rejecting strings). The third assertion can be thought of in two steps: first, that we can create a complement of a class being operated on by $\bigoplus$ by simply accepting one additional second-element for each first-element in the class (in other words, adding one to an even number makes an odd number and vice-versa); secondly, that the class Π_{k-1}^P is closed under this operation.

3.3 To Be Continued next class...