CS 221: Computational Complexity **Prof. Salil Vadhan**

Lecture Notes 13

March 10, 2010 **Scribe: James Williamson**

1 Randomized Reductions

We consider UNIQUE SAT, a promise problem

$$\begin{aligned} \mathrm{USAT}_Y &= \{\varphi \,|\, \varphi \text{ has exactly one satisfying assignment }\} \\ \mathrm{USAT}_N &= \{\varphi \,|\, \varphi \text{ is unsatisfiable }\} \end{aligned}$$

We can easily reduce USAT to SAT by mapping any formula to itself. Any formula that is a yes instance is in SAT and any no instance is not in SAT.

We now show that USAT reduces to SAT via a randomized reduction. Therefore if we are allowed randomness, USAT is no easier than SAT.

Theorem 1 *(Valiant-Vazirani)* SAT $\leq_r$ USAT*, where* $\leq_r$ *denotes a randomized Karp reduction. More specifically:* $\exists$ *a PPT algorithm* M *such that*

$$\varphi \in \mathrm{SAT} \Rightarrow \Pr[M(\varphi) \in \mathrm{USAT}_Y] \geq 1/8n$$

$$\varphi \notin \mathrm{SAT} \Rightarrow \Pr[M(\varphi) \in \mathrm{USAT}_N] = 1$$

n is the number of variables in ϕ.

Corollary 2 USAT $\in$ **prBPP** $\iff$ SAT $\in$ **BPP**

Proof: The idea is to use hashing to randomly remove satisfying assignments.

Definition 3 $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}^m\}$ *is a* pairwise independent family of hash functions *if* $\forall x_1 \neq x_2 \in \{0,1\}^n, y_1, y_2 \in \{0,1\}^m$

$$\Pr_{h \xleftarrow{R} \mathcal{H}}[h(x_1) = y_1 \wedge h(x_2) = y_2] = 1/2^{2m}$$

.

While a completely random hash function from $\{0,1\}^n$ to $\{0,1\}^m$ would require exponentially many random bits to generate and describe, it turns out that pairwise independent families can be generated using polynomially many random bits and can be evaluated efficiently:

Lemma 4 *For all* $n, m \in \mathbb{N}$, $\mathcal{H}_{n,m} = \{h_{A,b} : A \in \{0,1\}^{n\times m}, b \in \{0,1\}^m\}$ *is a pairwise independent family, where* $h_{A,b}(x) = Ax + b$*, and all arithmetic is modulo 2.*

To do the reduction from SAT to USAT: choose $m \xleftarrow{R} \{2, \ldots, n+1\}$, $A \xleftarrow{R} \{0,1\}^{n \times m}$, $b \xleftarrow{R} \{0,1\}^m$ (all uniformly at random). Then the mapping operates as follows:

$$\varphi(x) \mapsto \varphi'(x) = \varphi(x) \wedge (h_{A,b}(x) = 0^m)$$

If $x \notin$ SAT then $\varphi(x) = 0 \Rightarrow \varphi'(x) = 0$. Therefore the reduced formula is in the no instance of USAT with probability 1.

If $x \in$ SAT we show the following:

Claim 5 *If* $2^{m-2} \leq \left|\varphi^{-1}(1)\right| \leq 2^{m-1}$ *then* $\Pr\left[\varphi' \in \mathrm{USAT}_Y\right] \geq 1/8$

Proof of claim:

$$\begin{aligned}
\Pr\left[\varphi' \in \mathrm{USAT}_Y\right] &= \sum_{x \in \varphi^{-1}(1)} \Pr\left[x \text{ is a unique assignment to } \varphi'\right] \\
&\geq \sum_{x \in \varphi^{-1}(1)} \Pr\left[h(x) = 0\right] - \sum_{y \in \varphi^{-1}(1) \setminus \{x\}} \Pr\left[h(y) = h(x) = 0\right] \\
&\geq \left|\varphi^{-1}(1)\right| \left(1/2^m - \left|\varphi^{-1}(1)\right| \cdot 1/2^{2m}\right) \\
&= \frac{\left|\varphi^{-1}(1)\right|}{2^m} \cdot \left(1 - \frac{\left|\varphi^{-1}(1)\right|}{2^m}\right) \\
&\geq 1/4 \cdot (1/2) = 1/8
\end{aligned}$$

□

The result follows since m is chosen so that the inequality above holds. The bound on the probability follows easily. ■

2 Counting Complexity

The goal of this topic is to count the number of witnesses to problems in NP.

Definition 6 $f : \{0,1\}^* \to \mathbb{N}$ *is in* #**P** *if* $\exists$ *a polynomial* p *and a polynomial-time algorithm* M *such that for all* x,

$$f(x) = \#\{y \in \{0,1\}^{p(|x|)} \mid M(x,y) = 1\}.$$

2.1 Examples and Motivations

- #SAT.

  $$M(\varphi, y) = \begin{cases} 1 & \varphi(y) = 1 \\ 0 & \varphi(y) = 0 \end{cases}$$

  with $f(\varphi) = \left|\varphi^{-1}(1)\right|$. It is clearly as hard as deciding SAT.

- Examples of when there are in fact nice closed form formulas:

- Matrix-Tree Theorem: The number of spanning trees of a graph G with adjacency matrix A is given by $\det(L(G))$ where $L(G)$ is the *graph Laplacian* defined by

$$L(G) = \begin{pmatrix} d_1 & & 0 \\ & \ddots & \\ 0 & & d_n \end{pmatrix} - A$$

  and d_i is the degree of vertex i.
- The Fisher-Kestelyn-Tempotley Algorithm: Let G be a planar graph. Then using the embedding into the plane we can construct an efficiently computable signed, skew-symmetric version of the adjacency matrix M such that the number of pefect matchings of G is given by $\sqrt{\det(M)}$.

- Examples of natural problems from various disciplines that give rise to harder counting problems.
  - Networking: Given a connected graph G where each edge fails with probability p what is probability that G remains connected? For $p = 1/2$, then is given by

$$\frac{\#\text{ spanning subgraphs of } G}{2^{|E|}}$$

    So we need to be able to count the number of spanning subgraphs, which turns out to be #**P**-complete. (As a note this is solvable for any given value of p in polynomial time in the presence of an oracle for #**P**.)
  - Statistical Mechanics: Consider a monomer, dimer system represented by a graph G. Each pair adjacent of vertices can be occupied by a dimer and all other vertices can be represented by a monomer. At equilibrium this system follows the Gibbs distribution where the probability of a configuration σ is given by

$$\Pr[\sigma] = \frac{\mu^{\#\text{dimers}}}{Z(G,\mu)},$$

    where μ is a parameter (governed e.g. by the temperature of the system). By necessity

$$Z(G,\mu) = \sum_{sigma} \mu^{\#\text{dimers}(\sigma)}$$

    for the formula to make sense. This function is hard to compute. If we let $\mu = 1$ then $Z(G,1)$ is the number of matchings in G, another natural problem that #**P**-complete.
  - Artificial Intelligence: Consider a Bayes Net with n hidden variables each of which is 0 with probability $1/2$ independently. We want to guess the n hidden variables given values to m observed variables.

    One possible question we could ask is $\Pr[x_1 = 1 \mid y_1 = \cdots = y_m = 1]$. We know how the network is constructed so we can write $y_1 = \phi_1(x_1, \ldots, x_n)$ and $\phi = \phi_1 \wedge \cdots \wedge \phi_m$. Then this probability becomes

$$\frac{\#\text{ satisfying assignments to } \phi|_{x_1=1}}{\#\text{ satisfying assignments to } \phi}$$

    Computing this quantity can be shown to be computationally equivalent to $\#SAT$.

2.2 #P Complete Problems

1. #Circuit Satisfiability: We can do the following reduction from any counting problem with verifier M: $x \mapsto C_x(\cdot) = M(x, \cdot)$. The number of satisfying assignments to C_x exactly equals the the number of solutions to the original counting problem on instance x. Such reductions are called *parsimonious*. That is we have $f \leq g$ via a reduction R such that for all x, $g(R(x)) = f(x)$.

2. #3SAT, the standard reduction from Circuit Satisfiability to 3SAT is parsimonious since the added gates have values determined by the input values.

The counting analogues of all known natural **NP** complete problems are #**P** complete under a reduction that takes the form $f(x) = S(g(R(x)))$ where R, S are polynomial time computable and S is usually multiplication by a constant factor (also referred to as parsimonious reductions).

However, there are a number of #**P**-completeness results that seem to require non-parsimonious reductions, or even Cook reductions. For instance #DNF: The reduction runs as follows: For ϕ in 3CNF take $\phi \mapsto \neg\phi$ under R. Now for $g(\phi) = k$ we know that $f(\phi)$ must be equal to $2^n - k$ where n is the number of variables in ϕ.

This not only shows an example where S is not a constant factor, but also demonstrates that there are problems that are complete for #**P** where the underlying decision problem is easy (as satisfiability of DNF formulas is easy to test).