CSE 531: Computational Complexity I **Winter 2016**

Lecture 14: The Complexity of Counting II

Feb 19, 2016

Lecturer: Paul Beame *Scribe: Paul Beame*

1 $\#\mathsf{P}$ and $\#\mathsf{P}$-completeness

The following theorem which we started to prove last time shows that $\#\mathsf{P}$ and PP are natural analogues of each other.

Theorem 1.1. $\mathsf{P}^{\mathsf{PP}} = \mathsf{P}^{\#\mathsf{P}}$.

Proof. (Finish)
$\subseteq$: Last time. $\supseteq$: We show that any $\#\mathsf{P}$ function can be computed in P^{PP}. The idea is to use the PP algorithm in a binary search for the value of the output $f(x)$. Since the output of the function has only $p(|x|)$ bits for some polynomial p, it suffices to show that one can test "$f(x) > N$" for any integer N of polynomially many bits via a call to a PP oracle.

Let M be the polynomial-time TM and p be the polynomial bound such that $f(x) = \#\{y \in \{0,1\}^{p(|x|)} \mid M(x,y) = 1\}$. Define $M'(x, N, y')$ for N a length $p(|x|)$ binary string viewed as an integer and $|y'| = p(|x|) + 1$ by $M'(x, N, 0y) = M(x, y)$ and $M'(x, N, 1y) = 1$ iff $y \geq N$. Note that there will be precisely $2^{p(|x|)} - N$ strings $y' = 1y$ such that $M'(x, N, y')$ accepts. Therefore $f(x) > N$ if and only if there are $> N$ strings $y' = 0y$ such that $M'(x, N, y')$ for a total of $> 2^{p(|x|)}$ witness strings y' such that $M'(x, N, y')$ accepts, which is $> 1/2$ of all strings of the witness length. The PP oracle will be for the language defined by the machine M' with input (x, N) and length $p(n) + 1$. It will accept (x, N) precisely when $f(x) > N$. $\square$

Definition 1.2. *A function g is* $\#\mathsf{P}$-complete *iff*

1. $g \in \#\mathsf{P}$, *and*
2. *For all $f \in \#\mathsf{P}$, $f \in \mathsf{FP}^{g}$.*

Theorem 1.3. *$\#3SAT$ is $\#\mathsf{P}$-complete.*

Proof. Clearly $\#3SAT \in \#\mathsf{P}$. Let $f \in \#\mathsf{P}$. Then there is a polynomial-time TM M and p a polynomial size bound such that $f(x) = \#\{y \in \{0,1\}^{p(|x|)} \mid M(x,y) = 1\}$. The claim is that in polynomial time we can convert M and x to a 3CNF formula $\varphi_{M,x}$ such that the number of satisfying assignments for $\varphi_{M,x}$ is precisely the number of y such that $M(x,y) = 1$. Such a reduction that preserves the number of witnesses is called "parsimonious". It is not hard to see that the reduction in the proof of the Cook-Levin Theorem from M, x to the $CIRCUIT\text{-}SAT$ instance $C_{M,x}$ is indeed parsimonious. Moreover, the reduction from $CIRCUIT\text{-}SAT$ to CNF formulas with at most 3 variables per clause by adding extension variables for each gate is also parsimonious since the gate values are uniquely determined by the input values if all the clauses are satisfied.

The only part of the reduction to $3SAT$ that was not parsimonious was the part where we added extra variables to bring the clause size up from 1 or 2 to length 3. These added 1 or 2 extra variables (which could be re-used in all clauses). When we map clause $(u \vee v)$ to $(u \vee v \vee a)(x \vee y \vee \neg a)$ we also choose some 3CNF formula containing a that has precisely one satisfying assignment and add it to φ. For example $(a \vee b \vee c)(a \vee b \vee \neg c)(a \vee \neg b \vee c)(a \vee \neg b \vee \neg c)(\neg a \vee b \vee c)(\neg a \vee b \vee \neg c)(\neg a \vee \neg b \vee c)$. These 7 clauses rule out every assignment but the assignment that sets $a = b = c =$true. This results in a parsimonious reduction to $3SAT$ as required. $\square$

From this we can derive that $\#HAMILTONIAN\text{-}CYCLE$ is $\#$P-complete and by the proof from last class we obtain that $\#CYCLE$ is also $\#$P-complete so problems in P can have $\#$P-complete counting versions. One can also show that $\#2SAT$ is $\#$P-complete.

Determinant and Permanent The determinant of an $n \times n$ matrix $A = (a_{ij})$ is defined by

$$DET(A) = \sum_{\sigma \in S_n} (-1)^{\text{sgn}(\sigma)} \prod_{i=1}^{n} a_{i\sigma(i)},$$

where S_n is the set of permutations on $\{1, \ldots, n\}$ and sgn of permutation σ is the number of transpositions needed to produce σ. The $DET(A)$ is efficiently computable using Gaussian elimination. One can similarly define the *permanent* of matrix A by

$$PERM(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i\sigma(i)}.$$

The definition of $PERM(A)$ seems simpler than that of $DET(A)$, but it appears to be more complicated to compute.

Define $01PERM$ to be the problem of computing the permanent of an input matrix $A \in \{0,1\}^{n \times n}$. If A is the adjacency matrix of a bipartite graph with vertex sets U, V with $|U| = |V| = n$ then each permutation σ for which $\prod_{i=1}^{n} a_{i\sigma(i)} = 1$ corresponds to a perfect matching from U to V.

Therefore computing $01PERM$ is equivalent to counting the number of bipartite matchines.

Theorem 1.4 (Valiant)**.** *$01PERM$ is $\#$P-complete.*

Corollary 1.5. *$\#BIPARTITE\text{-}MATCHINGS$ is $\#$P-complete.*

Proof Sketch of the Theorem. Another way to think of an $n \times n$ 01-matrix is as the adjacency matrix of a directed graph with self-loops. For an arbitrary $n \times n$ matrix, we can view it as a weighted directed graph. For a directed graph G, a *cycle-cover of G* is a union of simple cycles of G that contains each vertex precisely once. For a weighted, directed graph G, with weight matrix A, we can view the permanent of A as the total weight of all cycle-covers of G, where the weight of a cycle-cover is the product of the weights of all its edges. This interpretation corresponds naturally to the representation of a permutation σ as a union of directed cycles. For example, if $\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 2 & 1 & 6 \end{pmatrix} \in S_6$ then σ can also be written in cycle form as $(1\,3\,5)(2\,4)(6)$ where the notation implies that each number in the group maps to the next and the last maps to the first. (See Figure 1.) Thus, for an unweighted graph G, PERM(G) is the number of cycle-covers of G.

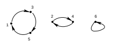

Figure 1: Directed graph corresponding to $(1\,3\,5)(2\,4)(6)$

The general idea si reduce $\#3SAT$ to $01PERM$ in two steps. Given any 3CNF formula φ, in the first step, we create a weighted directed graph G' (with small weights) such that $PERM(G')$ will represent the number of satisfying assignments for φ multiplied by some factor depending on the. the number of clauses in φ. In second step, we convert G' to an unweighted graph G such that $PERM(G') = PERM(G) \bmod M$, for some M with only have polynomially many bits. We will only sketch the general idea of the argument and will not give any details of the second step.

The construction of G' from φ is via gadgets. The VARIABLE gadget is shown in Figure 2. All the edges have unit weights. Notice that it contains one dotted edge for every occurrence of the variable in φ. Each dotted edge will be replaced by connecting two vertices of a subgraph which we described later. Any cycle-cover either contains all dotted edges corresponding to a positive occurrence (and all self-loops corresponding to negative occurrence) or vice versa.

The CLAUSE gadget is shown in Figure 3. It contains three dotted edges corresponding to three variables that occur in that clause. All the edges have unit weights. This gadget has the property that

1. in any cycle-cover, at least one of the dotted edges is *not* used, and
2. for any non-empty subset S of the dotted edges there is precisely one cycle-cover of the gadget that includes all dotted edges but those in S. (See Figure 4.)

Now, given any clause C and any literal x contained in it, there is a dotted edge (u, u') in the CLAUSE gadget for the literal and a dotted edge (v, v') in the appropriate side of VARIABLE gadget for the clause. These two dotted edges are replaced by an XOR gadget shown in Figure 5. This gadget has weights on some edges including some negative weights (all other edges have weight 1). The negative edges will be replaced in the final construction by weight $M-1$ since the result is being taken modulo M.

The XOR gadget has the property that the total contribution of all cycle-covers using none or both of (u, u') and (v, v') is 0. For cycle-covers using exactly one of the two, the gadget contributes a factor of 4.

There are $3m$ XOR gadgets. As a result, every satisfying assignment of truth values to φ will contribute 4^{3m} to the cycle-cover and every other assignment will contribute 0. The details are found in the text. $\square$

The fact that the permanent and determinant have similar representations, but one is computable in polynomial time and the other is $\#\mathsf{P}$-complete has been the inspiration for approaches to separating complexity classes. In particular, Mulmuley, in an approach called "geometric complexity theory" has suggested using extensions of the methods of algebraic geometry to prove that these two problems cannot have similar complexity.

2 Approximating $\#\mathsf{P}$ functions

Though it isn't clear how to compute the number of satisfying assignments exactly seems hard compared to BPP or PH, it turns out that we can *approximate* the number of satisfying assignments using properties of hash functions.

Theorem 2.1 (Stockmeyer's Theorem)**.** *Let $f \in \#\mathsf{P}$. For any $\varepsilon(n)$ that is $1/n^{O(1)}$ there is a function $g \in \mathsf{FP}^{\Sigma_2\mathsf{SAT}}$ such that for any x,*

$$(1-\varepsilon(|x|))f(x) \geq g(x) \leq (1+\varepsilon(|x|)f(x).$$

Proof. Let $f \in \#\mathsf{P}$ and M and p be the associated polynomial-time TM and bounding function. Let $m = p(|x|)$ and S be the set of $y \in \{0,1\}^m$ such that $M(x,y) = 1$. First, suppose that $|S| \leq 2^{k-1}$. Define $h : \{0,1\}^m \to \{0,1\}^k$ by $h(x) = A \cdot x$ for A an $m \times k$ Boolean matrix taken modulo 2.

Let $z \in S$. Then for a randomly chosen h, $\mathbb{P}_h[\exists y \neq z \in S.\ h(y) = h(z)] \leq |S|/2^k \leq 1/2$.

Then for a randomly and independently chosen $H = \{h_1, \ldots, h_k\}$,

$$\begin{aligned}\mathbb{P}_H[\exists y \neq z \in S\ \forall h_i \in H.\ h_i(y) = h_i(z)] &= \prod_{i=1}^{k} \mathbb{P}_{h_i}[\exists y \neq z \in S.\ h_i(y) = h_i(z)] \\ &\leq 2^{-k}\end{aligned}$$

It follows that

$$\mathbb{P}_H[\exists z\ \exists y \neq z \in S\ \forall h_i \in H.\ h_i(y) = h_i(z)] \leq |S|/2^k \leq 1/2.$$

Therefore

$$\mathbb{P}_H[\forall z\ \forall y \neq z \in S\ \exists h_i \in H.\ h_i(y) \neq h_i(z)] \geq 1/2.$$

Therefore if $|S| \leq 2^{k-1}$, then

$$\exists H \forall y, z \in S \bigvee_{i=1}^{k} (h_i(y) \neq h_i(z)).$$

On the other hand, suppose that $S \geq k2^k$. Suppose that the above sentence is true. Each $y \in S$ gets mapped to at least one unique pair $(i, h_i(y))$ that is not mapped to by any other element of S. There are at most $k2^k$ choices of such a value so if this is true then $|S| \leq k2^k$. Equivalently, if $f \leq 2^{k-1}$ then

$$\exists H \forall y, z \in \{0,1\}^m (M(x,y) = 1) \wedge (M(x,z) = 1) \wedge (y \neq z) \wedge \bigvee_{i=1}^{k} (h_i(y) \neq h_i(z)).$$

and it is false if $f > k2^k$.

In order to get a $1 \pm \varepsilon$ approximation to f, we first compute g which for some polynomial ℓ that counts the number of inputs $(y_1, \ldots, y_\ell) \in (\{0,1\}^m)^\ell$ such that $M(x, y_1) = \ldots = M(x, y_\ell) = 1$. We can then use the appropriate k to obtain the approximation as in the above construction for g. $\square$

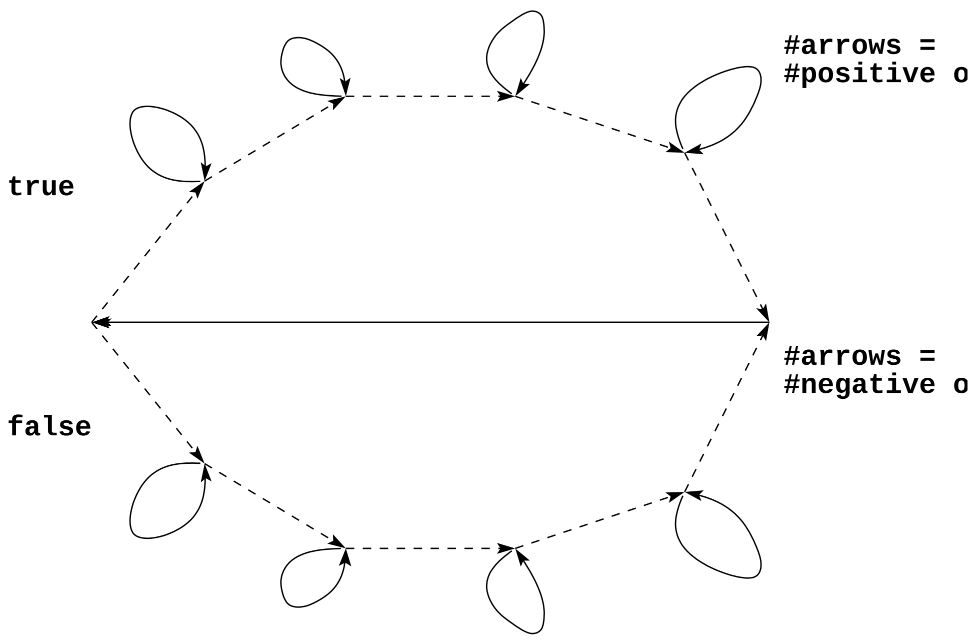

Figure 2: The VARIABLE gadget

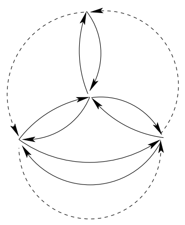

Figure 3: The CLAUSE gadget

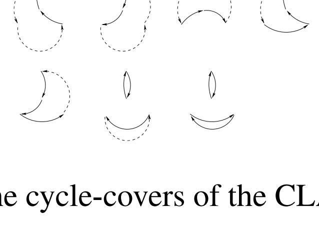

Figure 4: The cycle-covers of the CLAUSE gadget

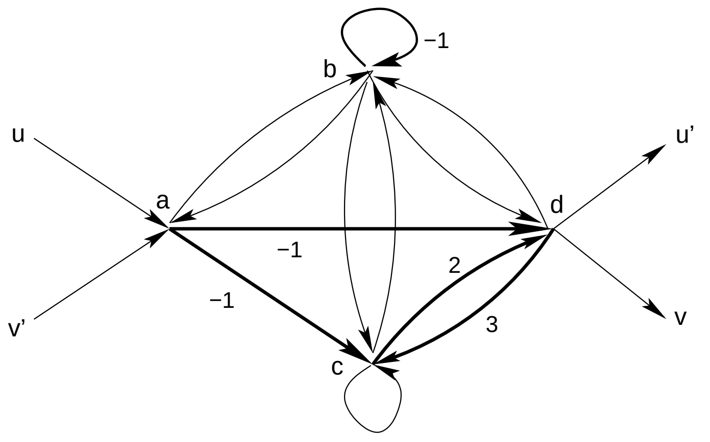

Figure 5: The XOR gadget