CAS CS 535: Complexity Theory

Lecturer: Mark Bun Fall 2026

Lecture Notes 19:

$\#\mathbf{P}$-Completeness, Permanent, Toda's Theorem

Reading.

- Arora-Barak § 17.2-17.4

Last time: Proof of Valiant-Vazirani, Intro to Counting

Definition 1. The complexity class $\mathbf{FP}$ consists of all functions $f : \{0,1\}^* \to \mathbb{N}$ such that f can be computed in deterministic polynomial time.

Definition 2. A function $f : \{0,1\}^* \to \mathbb{N}$ is in $\#\mathbf{P}$ (pronounced "sharp-P") if there exists a polynomial-time deterministic TM M and a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that

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

That is, for every x, the function $f(x)$ counts the number of witnesses u such that $M(x,u)$ accepts.

1 Reductions Between Function/Counting Problems

To study the relationships between different counting problems, we'll study two different ways of poly-time reducing between them.

Option 1: Parsimonious reductions. You should think of these as the function-evaluation analogs of Karp reductions (used to define $\mathbf{NP}$-completeness and so forth).

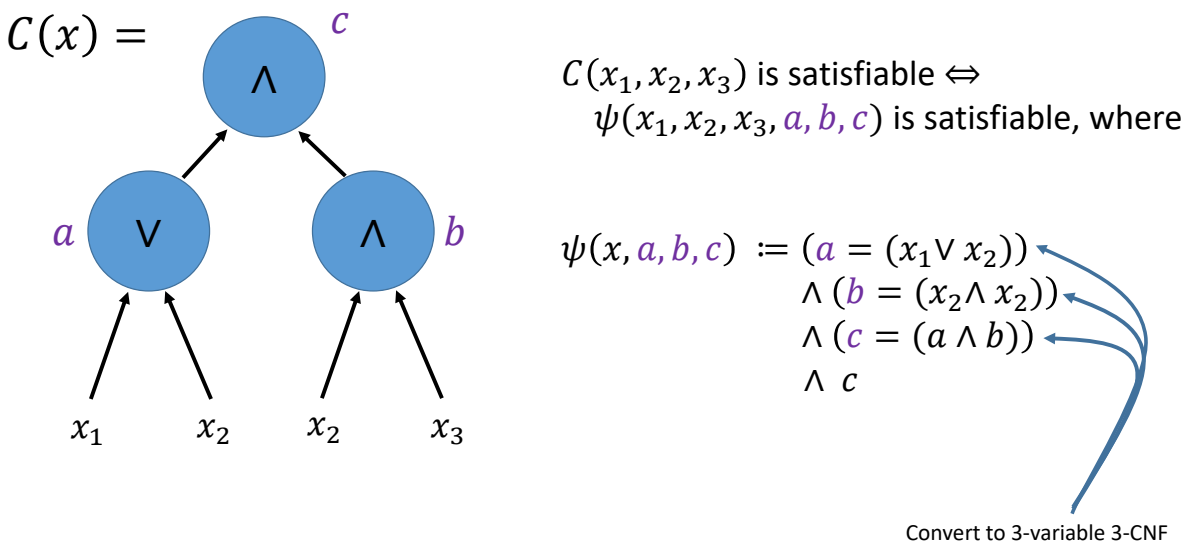
Definition 3. A *parsimonius reduction* R from f to g is a poly-time computable function such that for all $x \in \{0,1\}^*$, we have

$$f(x) = g(R(x)).$$

Interpretation: If computing g is easy, then computing f is also easy.

Example 4. $\#\mathsf{CKTSAT}$ parsimoniously reduces to $\#\mathsf{SAT}$. The reason is that the reduction from the decision version CKTSAT to SAT preserves the number of satisfying assignments.

Proof idea:



Example 5. The Cook-Levin Theorem is parsimonious, so there is a parsimonious reduction from every function $f \in \#\mathbf{P}$ to $\#\mathsf{SAT}$.

Theorem 6. $\#\mathsf{SAT}$ *is* $\#\mathbf{P}$*-complete under parsimonious reductions.*

Option 2: Oracle reductions. These are analogs of Cook reductions. (I.e., language A Cook-reduces to language B if $A \in \mathbf{P}^B$.) Oracle reductions are the more typical way to define $\#\mathbf{P}$-completeness.

We can generalize oracle access to a language to oracle access to a function as follows: We say that TM M has oracle access to function g if it has oracle access to the language $\{\langle x, i\rangle \mid (g(x))_i = 1\}$. Up to a polynomial in runtime, this is equivalent to having all of $g(x)$ appear on a work tape when x is queried.

Definition 7. Let $f, g : \{0,1\}^* \to \mathbb{N}$. Then f poly-time oracle reduces to g if $f \in \mathbf{FP}^g$.

Oracle reductions are more powerful than parsimonious reductions. Why should we study them? For one, there are natural examples of counting problems whose complexity is "obviously" equivalent, but which we do not believe are related by efficient parsimonious reductions. For example:

Example 8. $\#(\mathsf{CNF})\mathsf{SAT}$ poly-time (oracle) reduces to $\#\mathsf{DNFSAT}$. The reduction is as follows.

On input CNF formula $\varphi(x_1, \ldots, x_n)$:

1. Use de Morgan's laws to write $\neg\varphi$ as a DNF ψ.
2. Query $\#\mathsf{DNFSAT}$ on ψ, obtaining a count a.
3. Return $2^n - a$.

On the other hand, since the decision problem $\mathsf{DNFSAT} \in \mathbf{P}$, if $\#\mathsf{SAT}$ were to parsimonisouly reduce to $\#\mathsf{DNFSAT}$, we would have $\mathbf{P} = \mathbf{NP}$.

Definition 9. A function $g : \{0,1\}^* \to \mathbb{N}$ is $\#\mathbf{P}$-complete if $g \in \#\mathbf{P}$ and every function $f \in \#\mathbf{P}$ poly-time oracle reduces to g.

2 Valiant's Theorem: Permament is #P-Hard

Let's recall the definition of the determinant of a matrix $A \in \mathbb{R}^{n \times n}$:

$$\det(A) = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \cdot \prod_{i=1}^{n} A_{i,\sigma(i)}.$$

Here, S_n is the group of permutations over $[n]$, and for a permutation σ,

$$\operatorname{sgn}(\sigma) = \begin{cases} 1 \text{ if } \sigma \text{ has an even number of inversion}(i < j \text{ s.t. } \sigma(i) > \sigma(j)) \\ -1 \text{ otherwise.} \end{cases}$$

It is known that $\det \in \mathbf{FP}$, e.g., by doing Gaussian elimination until A is upper triangular, and then multiplying the diagonal entries.

The matrix *permanent* is defined by removing the sign terms:

$$\operatorname{perm}(A) = \sum_{\sigma \in S_n} \cdot \prod_{i=1}^{n} A_{i,\sigma(i)}.$$

Define the function $0/1\text{-perm}$ to capture the problem of computing the matrix permanent of $\{0,1\}$-valued matrices.

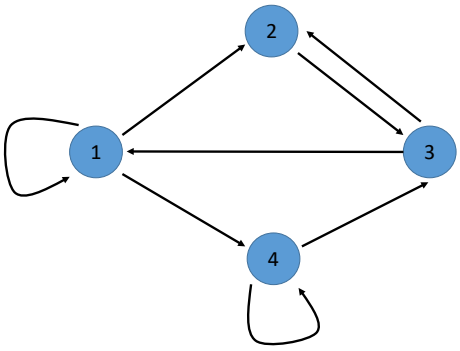
Theorem 10. $0/1\text{-perm}$ *is* $\#\mathbf{P}$*-complete.*

Proof sketch. As usual, there are two things to prove.

$0/1\text{-perm} \in \#\mathbf{P}$**:** Let us introduce a combinatorial view of the matrix permanent, in terms of *cycle covers*.

Definition 11. Let $G = ([n], E)$ be a directed graph, a permutation $\sigma : [n] \to [n]$ is a cycle cover for G if $(i, \sigma(i)) \in E$ for all $i \in [n]$.

Example 12. Consider the following digraph on 4 vertices.



The permutation $1 \mapsto 2, 2 \mapsto 3, 3 \mapsto 1, 4 \mapsto 4$ is a cycle cover.
The permutation $1 \mapsto 2, 2 \mapsto 3, 3 \mapsto 4, 4 \mapsto 1$ is not a cycle cover.

Given a matrix A, let G_A be the digraph whose adjacency matrix is A.

Lemma 13. $\mathrm{perm}(A) = \#$ cycle covers of G_A.

Proof. From the definition,

$$\begin{aligned}
\mathrm{perm}(A) &= \sum_{\sigma \in S_n} \prod_{i=1}^{n} A_{i,\sigma(i)} \\
&= \sum_{\sigma \in S_n} \begin{cases} 1 & \text{if } A_{i,\sigma(i)} = 1 \quad \forall i \in [n] \\ 0 & \text{otherwise} \end{cases} \\
&= \sum_{\sigma \in S_n} \begin{cases} 1 & \text{if } (i, \sigma(i)) \in E_A \quad \forall i \in [n] \\ 0 & \text{otherwise} \end{cases} \\
&= \#\text{ cycle covers of } G_A.
\end{aligned}$$

□

Since checking whether a permutation is a cycle cover can be done in poly-time, this implies that counting the number of cycle covers in a graph (and hence, computing the 0/1 permanent) is in $\#\mathbf{P}$.

Note that there's nothing really special about the 0/1 permanent here. This proof generalizes to characterize the integer permanent as summing the products of the weights along all possible weighted cycle covers.

Also, see page 347 in Arora-Barak for a different useful characterization of the permanent as counting perfect matchings in a bipartite graph.

0/1-perm **is $\#\mathbf{P}$-hard:** This is a quite involved proof involving some very clever gadget trickery. In broad strokes, the structure of the proof goes as follows.

1. Reduce the $\#\mathbf{P}$-complete problem $\#\mathsf{SAT}$ to integer perm. Idea: Map a CNF formula φ to a graph G such that assignments to φ correspond to (structured sets of) cycle covers. Satisfying assignments will all induce covers of weight k (for some parameter k depending on φ), while unsatisfying assignments have weight 0.
2. Reduce integer perm to $\{-1, 0, 1\}$-perm by replacing weighted edges with sets of parallel paths.
3. Reduce $\{-1, 0, 1\}$-perm to $\{0, 1\}$-perm using modular arithmetic tricks.

□

3 PP and $\oplus\mathbf{P}$

Recall that $\#\mathbf{P}$ is the set of functions f such that there is polynomial-time deterministic TM M and a polynomial $p : \mathbb{N} \to \mathbb{N}$ such that for all x,

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

Any function evaluation problem naturally gives rise to decision problems that correspond to computing individual bits of that function. We can further specialize our attention to the most significant bit and least significant bit of f, which have natural interpretations.

Definition 14. A language $L \in \mathbf{PP}$ if there exists a poly-time deterministic TM M and polynomial p such that

$$x \in L \iff \#\{u \in \{0,1\}^{p(|x|)} \mid M(x,u) = 1\} \geq \frac{1}{2} \cdot 2^{p(n)}.$$

Equivalently: $L \in \mathbf{PP}$ if there is a poly-time probabilistic TM M such that

$$x \in L \iff \Pr[M(x) = 1] \geq \frac{1}{2}.$$

The canonical $\mathbf{PP}$-complete problem is MAJSAT: $\varphi \in \mathsf{MAJSAT} \iff$ the majority of assignments x satisfy φ.

Lemma 15. $\mathbf{P}^{\mathbf{PP}} = \mathbf{P}^{\#\mathbf{P}}$.

Proof. It's enough to show that $\#\mathbf{P} \subseteq \mathbf{FP}^{\mathbf{PP}}$.

To see this, let $f \in \#\mathbf{P}$, so $f(x) = \#\{u \in \{0,1\}^{p(|x|)} \mid M(x,u) = 1\}$ for some machine M. We describe how to compute f in poly-time on a $\mathbf{PP}$-oracle TM as follows. Let $k = p(|x|)$. Let M be the following TM taking as input x together with $N \in \{0, 1, \ldots, 2^k\}$, as well as certificates $b \in \{0,1\}, u \in \{0,1\}^k$.

On input $\langle x, N\rangle$, certificate $\langle b, u\rangle$:

If $b = 0$: Output $M(x,u)$

If $b = 1$: Output 1 iff $u < N$, and 0 otherwise.

Observe that for every x, we have $\#\{\langle b,u\rangle \in \{0,1\}^{k+1} \mid M(\langle x,N\rangle, \langle b,u\rangle) = 1\} = f(x) + N$.

Using a $\mathbf{PP}$ oracle, we can thus compare $f(x) + N \stackrel{?}{\geq} \frac{1}{2} \cdot 2^{k+1} = 2^k$ for any N. Now we can binary search over N to find the smallest number N^* for which $f(x) + N^* \geq 2^k$, i.e., $f(x) + N^* = 2^k$, and output the answer $f(x) = 2^k - N^*$. □

Instead of asking for the most significant bit in a counting problem, one can also ask for the least significant bit.

Definition 16. A language $L \in \oplus\mathbf{P}$ if there exists a poly-time deterministic TM M and polynomial p such that

$$x \in L \iff \#\{u \in \{0,1\}^{p(|x|)} \mid M(x,u) = 1\} \text{ is odd.}$$

The canonical $\oplus\mathbf{P}$-complete problem is $\oplus$SAT: $\varphi \in \oplus\mathsf{SAT} \iff \#\varphi$ is odd.

We do not yet even know whether $\mathbf{NP} \subseteq \mathbf{P}^{\oplus\mathbf{P}}$, but we do know from Valiant-Vazirani that $\mathbf{NP} \subseteq \mathbf{RP}^{\oplus\mathbf{P}}$.

4 Toda's Theorem

Alternation and counting give two ways of generalizing the class $\mathbf{NP}$ which, at first glance, may seem incomparable in power. In 1991, Seinosuke Toda proved the stunning result that counting is, in fact, at least as powerful as alternation.

Theorem 17 (Toda's "First" Theorem)**.** $\mathbf{PH} \subseteq \mathbf{BPP}^{\oplus\mathbf{P}}$.

Theorem 18 (Toda's "Second" Theorem)**.** $\mathbf{BPP}^{\oplus\mathbf{P}} \subseteq \mathbf{P}^{\#\mathbf{P}}$.

Corollary 19 ("Toda's Theorem")**.** $\mathbf{PH} \subseteq \mathbf{P}^{\#\mathbf{P}}$.

I'll sketch a proof of the "First Theorem" following Fortnow's note, "A Simple Proof of Toda's Theorem." The proof makes use of the following three facts:

1. Valiant-Vazirani: There is a poly-time randomized reduction A such that

$$\varphi \in \mathsf{SAT} \implies \Pr[\#A(\varphi) = 1] \geq \frac{1}{8n} \implies \Pr[A(\varphi) \in \oplus\mathsf{SAT}] \geq \frac{1}{8n}$$
$$\varphi \notin \mathsf{SAT} \implies \Pr[\#A(\varphi) = 0] = 1 \implies \Pr[A(\varphi) \notin \oplus\mathsf{SAT}] = 1.$$

   Corollary 20. $\mathbf{NP} \subseteq \mathbf{RP}^{\oplus\mathsf{SAT}} \subseteq \mathbf{BPP}^{\oplus\mathbf{P}}$. *(Moreover, this proof relativizes.)*

2. $\oplus\mathbf{P}^{\oplus\mathbf{P}} = \oplus\mathbf{P}$. (The proof uses some counting mod 2 tricks. I'll leave it to you as a bonus problem.)

3. If $\mathbf{NP} \subseteq \mathbf{BPP}$, then $\mathbf{PH} \subseteq \mathbf{BPP}$. (This generalizes the inductive proof that $\mathbf{NP} \subseteq \mathbf{P} \implies \mathbf{PH} \subseteq \mathbf{P}$.) Moreover, this proof relativizes.

Proof of Toda 1. Using the fact that the Corollary to Fact 1 implies relativizes, we have

$$\mathbf{NP}^{\oplus\mathbf{P}} \subseteq (\mathbf{BPP}^{\oplus\mathbf{P}})^{\oplus\mathbf{P}} = \mathbf{BPP}^{(\oplus\mathbf{P}^{\oplus\mathbf{P}})}.$$

The equality holds because a BPP machine can make its "outer" $\oplus\mathbf{P}$ queries via its "inner" $\oplus\mathbf{P}$ oracle.

Now Fact 2 implies this latter class is just $\mathbf{BPP}^{\oplus\mathbf{P}}$. So using the relativizing version of Fact 3, we get $\mathbf{PH} \subseteq \mathbf{PH}^{\oplus\mathbf{P}} \subseteq \mathbf{BPP}^{\oplus\mathbf{P}}$. □

Now let's sketch a proof Toda's "Second Theorem." The proof relies on a construction of a "modulus-amplifying polynomial" defined as follows.

Definition 21. An integer polynomial $p(s)$ is k-modulus amplifying if

$$s \equiv 0 \pmod 2 \implies p(s) \equiv 0 \pmod{2^k}$$
$$s \equiv -1 \pmod 2 \implies p(s) \equiv -1 \pmod{2^k}.$$

There's a very simple, efficiently computable, construction of such polynomials as follows. Define $g(s) = 3s^4 + 4s^3$. One can check, inductively, that $p(s) = g(g(\ldots g(s)))$ obtained by composing g a total of $\log k$ times does the trick.

The second idea is that, for any polynomial p with natural coefficients, we can convert any Boolean formula φ into another formula φ' such that $\#\varphi' = p(\#\varphi)$. This is because we can "multiply" counts of satisfying assignments:

$$\#(\varphi_1(x) \wedge \varphi_1(y)) = \#\varphi_1 \cdot \#\varphi_2$$

as well as add them:

$$\#((\varphi_1(x) \wedge y = 0) \vee (\varphi_2(y) \wedge x = 0)) = \#\varphi_1 + \#\varphi_2$$

Proof Sketch of Toda 2. One can show that $\oplus\mathsf{SAT}$ is complete for $\mathbf{BPP}^{\oplus\mathbf{P}}$ under *randomized* reductions with two-sided error. That is, for every $L \in \mathbf{BPP}^{\oplus\mathbf{P}}$ there's a poly-time algorithm $A(x; r)$ mapping $x \mapsto \varphi_{x,r}$ such that

$$x \in L \implies \Pr_r[\#\varphi_{x,r} \equiv -1 \pmod 2] \geq 2/3,$$
$$x \notin L \implies \Pr_r[\#\varphi_{x,r} \equiv 0 \pmod 2] \geq 2/3.$$

Let $k = |r| + 1$. Then using a k-modulus amplifying polynomial, we can, in poly-time further map $\varphi_{x,r} \mapsto \varphi'_{x,r}$ such that

$$x \in L \implies \Pr_r[\#\varphi'_{x,r} \equiv -1 \pmod{2^k}] \geq 2/3 \implies \sum_r \#\varphi'_{x,r} \in \left[-2^{|r|}, -\frac{2}{3}2^{|r|}\right] \pmod{2^k}$$

$$x \notin L \implies \Pr_r[\#\varphi'_{x,r} \equiv 0 \pmod{2^k}] \geq 2/3 \implies \sum_r \#\varphi'_{x,r} \in \left[-\frac{1}{3}2^{|r|}, 0\right] \pmod{2^k}.$$

Since these ranges are disjoint, they can be distinguished using one $\#\mathbf{P}$ oracle call to count the total number of satisfying assignments to all possible formulas $\varphi'_{x,r}$ that can arise in the reduction. □