

Computational Complexity Theory - Course Notes

CCT Team

October 25, 2006

12 Lecture 12, 6/10-2006

12.1 Decision Problems vs General Tasks

Examples of general tasks:

- Given an **NP**-type problem, find a solution: Given $L' \in \mathbf{P}$ and x find a short y so that $\langle x, y \rangle \in L'$. If $L = \{x \mid \exists y : \langle x, y \rangle \in L'\}$ is **NPC** then L captures the difficulty of the search problem (by the binary search trick). For other problems in **NP** it is not so clear. Example: For Nash Equilibria, where a solution is always known to exist, it is not clear how to formulate a decision that captures the difficulty of the task of finding the solution.
- Approximation algorithms (optimization, integration) are similar to the **NP**-type problems.

- Promise Problems:

Example: UNIQUE-FORMULA SAT

From FORMULA SAT we can derive UNIQUE-FORMULA SAT: Let U be the set of encodings of formulae with 0 or 1 satisfying assignments, and let L be the set of encodings of formulae with 1 satisfying assignment. Given $f \in U$ decide $f \in L$, i.e. given that f has 0 or 1 satisfying assignments, does it have one ?

12.2 Promise Problems

Def. 1 | Promise Problem

A Promise Problem is given by $U \subseteq \{0,1\}^*$ and $L \subseteq U$. An algorithm solves (U, L) if $\forall x \in U$ the algorithm correctly decides membership in L .

Def. 2 | Promise classes

Using the definition above we naturally define the promise classes **prP**, **prNP**, **prEXP**, ...

Def. 3 | Oracle machine $M^{(U,L)}$ with a Promise Problem

If the question $\notin U$, M does not have to keep the promise and the oracle gives an arbitrary answer.

Thm. 4 | Valiant-Vazirani

NP \neq **RP** \Rightarrow UNIQUE-FORMULA SAT \notin **prP**

Proof. The proof is contrapositive by showing FORMULA SAT \in **RP**^{UFS}.

Let $f \in$ FORMULA SAT with n variables. Pick $(n+1)n + (n+1) = n^2 + 2n + 1$ random bits for the $(n+1) \times n$ matrix A and the $(n+1)$ -column vector b . Let $A^{(i)}$ to be $i \times n$ matrix consisting of the first i rows A and let $b^{(i)}$ be the i -column vector consisting of the first i entries of b . Using matrix algebra over GF(2), i.e. summation is XOR and multiplication is AND, we define the hash-functions $g_i : \{0, 1\}^n \rightarrow \{0, 1\}$ by

$$g_i(x_0, x_1, \dots, x_n) = \left[A^{(i)} x^T \oplus b^{(i)} \equiv 0^i \right] \quad 1 \leq i \leq n+1$$

Obs: $f \wedge g_0 \wedge g_1 \wedge \dots \wedge g_i \equiv f \wedge g_i$.

Obs: g_i has a small $\{\cdot, \oplus\}$ formula of depth $O(\log n)$, and using the $a \oplus b = (a \wedge \neg b) \vee (\neg a \wedge b)$ the corresponding $\{\wedge, \vee, \neg\}$ formula is only polynomially larger.

The algorithm is as follows:

- Pick A, b at random.
- Feed the oracle $f \wedge g_1, f \wedge g_2, f \wedge g_3, \dots$.
- If the oracle says yes to $f \wedge g_i$, do the binary search trick to find an assignment, accept if it is satisfying and continue if it is not.

We need to show that if f is satisfiable, then with probability $\geq \frac{1}{8}$ one $f \wedge g_i$ has a unique satisfying assignment. We will then just need to repeat k times such that $(\frac{7}{8})^k < \frac{1}{2}$.

Let $S \subseteq \{0, 1\}^n$ be the satisfying assignments. By assumption $|S| \geq 1$. Let

$$f_i(x) = A^{(i)} x^T \oplus b^{(i)}$$

The probability, for a particular i , that $f \wedge g_i$ has a unique satisfying assignment is

$$\begin{aligned} & \Pr [\exists x \in S : f_i(x) = 0^i \wedge \forall y \in S - \{x\} : f_i(y) \neq 0^i] \\ \stackrel{\text{disjoint events}}{=} & \sum_{x \in S} \Pr [f_i(x) = 0^i \wedge \forall y \in S - \{x\} : f_i(y) \neq 0^i] \\ = & \sum_{x \in S} \Pr [f_i(x) = 0^i] \cdot \Pr [\forall y \in S - \{x\} : f_i(y) \neq 0^i \mid f_i(x) = 0^i] \\ = & \sum_{x \in S} \Pr [f_i(x) = 0^i] \cdot \left(1 - \sum_{y \in S - \{x\}} \Pr [f_i(y) = 0^i \mid f_i(x) = 0^i] \right) \\ \geq & \sum_{x \in S} \frac{1}{2^i} \left(1 - \sum_{y \in S - \{x\}} \frac{1}{2^i} \right) \\ = & |S| \frac{1}{2^i} \left(1 - (|S| - 1) \frac{1}{2^i} \right) \end{aligned}$$

If $|S| \in [2^{i-2}, 2^{i-1}]$ we have

$$|S| \frac{1}{2^i} \geq \frac{1}{4} \quad \text{and} \quad 1 - (|S| - 1) \frac{1}{2^i} \geq \frac{1}{2}$$

and therefore

$$\Pr [\exists x \in S : f_i(x) = 0^i \wedge \forall y \in S - \{x\} : f_i(y) \neq 0^i] \geq \frac{1}{8}$$

Since $1 \leq i \leq n + 1$ we have that probability that one $f \wedge g_i$ has a unique satisfying assignment is $\geq \frac{1}{8}$. \square

Note: $\mathbf{NP} \neq \mathbf{P} \Rightarrow \text{UNIQUE-FORMULA SAT} \notin \mathbf{prP}$ is open.

Note: In the original reduction $\mathbf{NP} \ni L \leq \text{CIRCUIT SAT}$ the circuit has 0 or 1 satisfying assignments, and the reduction $\text{CIRCUIT SAT} \leq \text{SAT}$ preserves the number of satisfying assignments. (See <http://www.daimi.au.dk/dKS/np.pdf>.)

Exercise: Show that if the polynomial hierarchy does not collapse, then $\mathbf{NP} \neq \mathbf{RP}$.

Exercise: Show $\mathbf{NP} \neq \mathbf{RP} \Rightarrow \text{UNIQUE-CNF-SAT} \notin \mathbf{prP}$.

12.3 BPP

Def. 5 | BPP

$L \in \mathbf{BPP} \Leftrightarrow \exists L' \in \mathbf{P}$, poly p :

$$\begin{aligned} x \in L &\Rightarrow \text{random } y \in \{0, 1\}^{p(|x|)} : \Pr [\langle x, y \rangle \in L'] \geq \frac{2}{3} \\ x \notin L &\Rightarrow \text{random } y \in \{0, 1\}^{p(|x|)} : \Pr [\langle x, y \rangle \in L'] \leq \frac{1}{3} \end{aligned}$$

Note: No complete problem for **BPP** is known.

Def. 6 | prBPP

$(U, L) \in \mathbf{prBPP} \Leftrightarrow \exists L' \in \mathbf{P}$, poly p :

$$\begin{aligned} x \in L &\Rightarrow \text{random } y \in \{0, 1\}^{p(|x|)} : \Pr [\langle x, y \rangle \in L'] \geq \frac{2}{3} \\ x \notin U - L &\Rightarrow \text{random } y \in \{0, 1\}^{p(|x|)} : \Pr [\langle x, y \rangle \in L'] \leq \frac{1}{3} \end{aligned}$$

Note: If (U, L) is complete for **prBPP** then

$$\begin{aligned} (U, L) \in \mathbf{prP} &\Leftrightarrow \text{Integration task can be solved by} \\ &\quad \text{a polynomial time deterministic algorithm} \\ &\Leftrightarrow \text{Monte Carlo integration can be derandomized in general} \end{aligned}$$

Def. 7 | Integration task

Given a circuit $C : \{0, 1\}^r \rightarrow \{0, 1\}$ and 1^k , estimate $\Pr_x [C(x) = 1]$ within additive error $\frac{1}{k}$.

Note: Integration task has a randomized algorithm solving the task with probability $\geq \frac{9}{10}$.

Thm. 8 | Complete Problem for **prBPP**

The promise problem (U, L) given by

$$U = \left\{ C \mid \text{either } \Pr_y [C(y) = 1] \geq \frac{2}{3} \text{ or } \Pr_y [C(y) = 1] \leq \frac{1}{3} \right\}$$
$$L = \left\{ C \mid \Pr_y [C(y) = 1] \geq \frac{2}{3} \right\}$$

is hard for **prBPP** by the tableau method, and $(U, L) \in \mathbf{prBPP}$ by the definition of Integration task.