CAS CS 535: Complexity Theory

Lecturer: Mark Bun Fall 2023

Lecture Notes 17:

Finish $\mathrm{BPP} \subseteq \mathrm{PH}$, Promise Problems, Randomized Reductions

Reading.

- Arora-Barak § 7.4-7.5

Last time: $\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{coRP}$, Error Reduction, $\mathbf{BPP} \subseteq \mathbf{P}_{/\mathbf{poly}}$

1 BPP and the Polynomial Hierarchy

It seems unlikely that $\mathbf{NP} \subseteq \mathbf{BPP}$, but what about the other direction? We know that $\mathbf{RP} \subseteq \mathbf{NP}$, but the power of two-sided error doesn't immediately appear compatible with just nondeterministic guessing. We don't yet know whether $\mathbf{BPP}$ is contained in $\mathbf{NP}$, but we can at least place it in the polynomial hierarchy.

Theorem 1 (Sipser-Gács-Lautemann)**.** $\mathbf{BPP} \subseteq \mathbf{\Sigma}_2^p \cap \mathbf{\Pi}_2^p$.

Proof. Since $\mathbf{BPP}$ is closed under complement, it suffices to show that $\mathbf{BPP} \subseteq \mathbf{\Sigma}_2^p$. Using error reduction, if $L \in \mathbf{BPP}$, then there is a poly-time TM M such that for all $x \in \{0,1\}^*$,

$$\Pr_{r \in \{0,1\}^m}[M(x,r) = L(x)] \geq 1 - 2^{-n}$$

for some $m = \mathrm{poly}(n)$.

For each $x \in \{0,1\}^n$, define the set

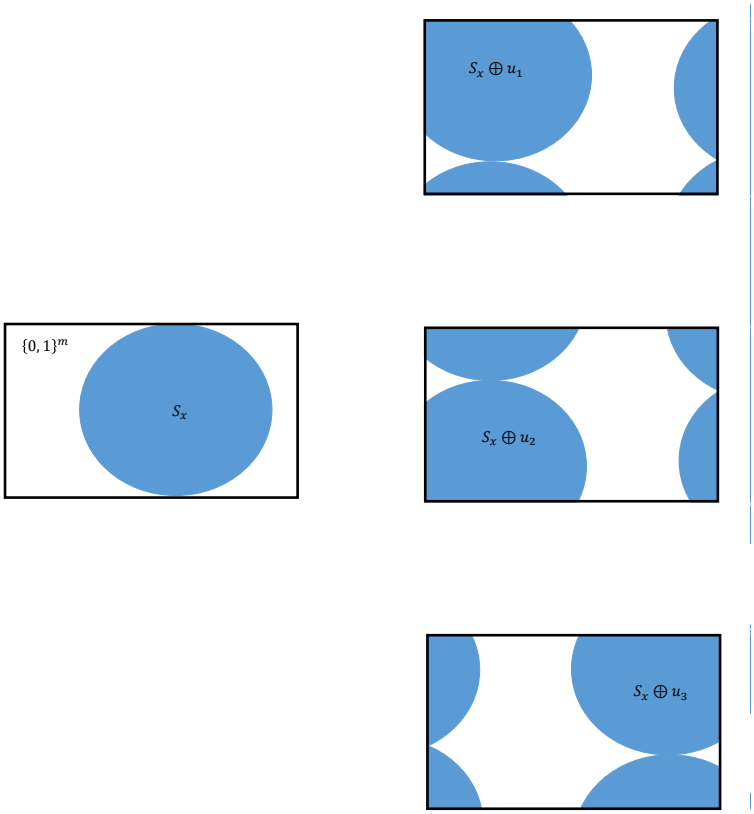
$$S_x = \{r \mid M(x,r) = 1\}$$

to be the set of coin tosses that cause M to accept input x. Then we have

$$x \in L \implies \frac{|S_x|}{2^m} \geq (1 - 2^{-n})$$
$$x \notin L \implies \frac{|S_x|}{2^m} \leq 2^{-n}$$

Our goal will be to use two quantifiers to distinguish between these cases, i.e., to check whether S_x is huge (almost everything) or tiny (almost nothing).

The idea is as follows. If S_x, then there will *exist* a small number of "additive shifts" of S_x whose union cover *all* of the space $\{0,1\}^m$.



Meanwhile, if S_x is tiny, then every small number of shifts will still fail to cover the whole space. Here are the details. First, let us define what we mean by additive shifts.

Definition 2. For $S \subseteq \{0,1\}^m, u \in \{0,1\}^m$, let $S \oplus u = \{z \oplus u \mid z \in S\}$, where $\oplus$ denotes the bitwise XOR.

Let $k = 2m/n = \text{poly}(n)$. To distinguish our two cases, our goal is to prove the following two lemmas.

Lemma 3. *If $|S_x|/2^m \geq 1 - 2^{-n}$, then there exist shifts $u_1, \ldots, u_k \in \{0,1\}^m$ such that $\bigcup_{i=1}^k (S_x \oplus u_i) = \{0,1\}^m$.*

Lemma 4. *If $|S_x|/2^m \leq 2^{-n}$, then for all shifts $u_1, \ldots, u_k \in \{0,1\}^m$, we have $\bigcup_{i=1}^k (S_x \oplus u_i) \subsetneq \{0,1\}^m$.*

Assuming these lemmas, let us now see how we can distinguish our two cases using two quantifiers.

Starting with the YES case, we have by Lemma 3 that

$$\begin{aligned}
x \in L &\implies |S_x|/2^m \geq 1 - 2^{-n} \\
&\implies \exists u_1, \ldots, u_k \in \{0,1\}^m \text{ s.t. } \bigcup_{i=1}^{k} (S_x \oplus u_i) = \{0,1\}^m \\
&\implies \exists u_1, \ldots, u_k \in \{0,1\}^m \forall r \in \{0,1\}^m \qquad r \in \bigcup_{i=1}^{k} (S_x \oplus u_i) \\
&\implies \exists u_1, \ldots, u_k \in \{0,1\}^m \forall r \in \{0,1\}^m \quad (\exists i \in [k]\ r \oplus u_i \in S_x) \\
&\implies \exists u_1, \ldots, u_k \in \{0,1\}^m \forall r \in \{0,1\}^m (\exists i \in [k]\ M(x, r \oplus u_i) = 1).
\end{aligned}$$

Similarly, in the NO case, we have by Lemma 4 that

$$x \notin L \implies \neg \exists u_1, \ldots, u_k \in \{0,1\}^m \forall r \in \{0,1\}^m (\exists i \in [k]\ M(x, r \oplus u_i) = 1).$$

Thus, we can distinguish the two cases using a poly-length round of existential guessing, followed by a poly-length round of universal guessing, and evaluation of the poly-time computable predicate "$\exists i \in [k]\ M(x, r \oplus u_i) = 1$." □

Now let's prove the lemmas.

Proof of Lemma 4. It suffices to show that if $|S|/2^m \leq 2^{-n}$, then for all sequences of k bitstrings $u_1, \ldots, u_k \in \{0,1\}^m$, where $k = 2m/n$, we have that

$$\left| \bigcup_{i=1}^{k} S \oplus u_i \right| < 2^m.$$

to do this, we use the union bound to upper bound the left hand side via

$$\sum_{i=1}^{k} |S \oplus u_i| = k|S| = \frac{2m}{n} \cdot 2^{-n} \cdot 2^m < 2^m$$

for sufficiently large n, since $m = \text{poly}(n)$. □

Proof of Lemma 3. Now we want to show that if $|S|/2^m \geq 1 - 2^{-n}$, there exist $u_1, \ldots, u_k$ such that $\bigcup_{i=1}^{k} S \oplus u_i = \{0,1\}^m$. We'll do this via the probabilistic method, showing that a random choice of the u's works with positive probability.

Let us choose $u_1, \ldots, u_k \in \{0,1\}^m$ uniformly at random. It now suffices to show that

$$\Pr_u \left[\bigcup_{i=1}^{k} S \oplus u_i = \{0,1\}^m \right] > 0$$

or equivalently, that

$$\Pr \left[\exists r \in \{0,1\}^m \text{ s.t. } r \notin \bigcup_{i=1}^{k} S \oplus u_i \right] < 1.$$

We bound the left hand side using a union bound by

$$\begin{aligned}
\sum_{r\in\{0,1\}^m} \Pr\left[r \notin \bigcup_{i=1}^{k} S \oplus u_i\right] &= \sum_{r\in\{0,1\}^m} \Pr[r \notin S \oplus u_1] \cdot \Pr[r \notin S \oplus u_2] \cdot \ldots \cdot \Pr[r \notin S \oplus u_k] \quad \text{by independence} \\
&= \sum_{r\in\{0,1\}^m} \prod_{i=1}^{k} \Pr[r \notin S \oplus u_i] \\
&\leq \sum_{r\in\{0,1\}^m} (2^{-n})^k \\
&\leq 2^m \cdot 2^{-n\cdot 2m/n} \\
&= 2^{-m} < 1.
\end{aligned}$$

□

2 BPP-Complete Problems?

One notion that's been conspicuously missing from our treatment of randomized computation is *complete problems*. Let's see what happens if we try to identify a $\mathbf{BPP}$-complete problem. Consider the following candidate (inspired, e.g., by our first $\mathbf{NP}$-complete problem).

$$\mathsf{BPA}_{\mathsf{TM}} = \{\langle M, x, 1^t\rangle \mid \text{ PTM } M \text{ accepts } x \text{ within } t \text{ steps w.p. } \geq 2/3\}.$$

One can show that this is $\mathbf{BPP}$-hard under poly-time reductions in exactly the same way we did it for $\mathbf{NP}$. The problem is that it isn't known whether this language is in $\mathbf{BPP}$! To illustrate the difficulty, consider the natural candidate algorithm for solving this problem on a PTM:

On input $\langle M, x, 1^t\rangle$:

Simulate M on x for t steps, and accept iff it accepts.

This works fine as long as, for every input x, the acceptance probability of M after t steps is either at least $2/3$ or at most $1/3$. But what if it's, say $1/2$? Then our algorithm should reject at least $2/3$ of the time, but only rejects $1/2$ of the time. Now you might imagine fixing this with repetitions...but what if M's acceptance probability is, say $2/3 - 2^{-|x|}$?

This issue actually seems fundamental, and in fact, there are no known $\mathbf{BPP}$-complete problems. The workaround to this is to relax our definition of $\mathbf{BPP}$, and indeed of languages themselves, to "promise problems."

Definition 5. A promise problem is a pair of sets (Π_Y, Π_N) such that $\Pi_Y, \Pi_N \subseteq \{0,1\}^*$ and $\Pi_Y \cap \Pi_N = \emptyset$.

The way you should interpret this is that Π_Y is the set of "YES" instances to a problem and Π_N is the set of "NO" instances. Anything outside of $\Pi_Y \cup \Pi_N$ is a "don't care" instance. An equivalent way of thinking about this is that, to solve a promise problem (Π_Y, Π_N), you (an algorithm) are given an input x that is *promised* to be in either Π_Y or Π_N, and your job is to determine which is the case. This notion is most interesting when it's hard (possibly even undecidable) to determine whether the promise holds for an input or not.

Example 6. Consider the unambiguous satisfiability problem USAT defined as follows: Given a Boolean formula φ that is promised to be either unsatisfiable or to have exactly one satisfiying assignment, determine which is the case.

To formalize this as a promise problem, we take

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

A language L is a special case of a promise problem where $\Pi_Y = L, \Pi_N = \overline{L}$, and there are no "don't care"s.

Definition 7. A promise problem (Π_Y, Π_N) is in $\mathbf{PromiseBPP}$ if there exists a polynomial-time PTM M such that

$$\begin{aligned}x \in \Pi_Y &\implies \Pr[M(x) = 1] \geq 2/3\\ x \in \Pi_N &\implies \Pr[M(x) = 1] \leq 1/3.\end{aligned}$$

With this definition in hand, we can now successfully describe $\mathbf{PromiseBPP}$-complete problems. One such problem is obtained by modifying $\mathsf{BPA}_{\mathsf{TM}}$ to include the promise that the TM's acceptance probability is bounded away from $1/2$. A similar problem that's usually easier to work with is the "Circuit Acceptance Probability Problem", which roughly captures the following question: Given a circuit C, what is the probability that C accepts a random input x?

Definition 8. The Circuit Acceptance Probability Problem (CAPP) is the promise problem defined as follows.

$$\begin{aligned}\Pi_Y &= \{\langle C, p\rangle \mid \Pr_x[C(x) = 1] \geq p + 1/10\}\\ \Pi_N &= \{\langle C, p\rangle \mid \Pr_x[C(x) = 1] \leq p - 1/10\}.\end{aligned}$$

3 Randomized Reductions and Valiant-Vazirani

(Deterministic) poly-time reductions help us study questions like "$\mathbf{NP} \stackrel{?}{=} \mathbf{P}$" and "$\mathbf{BPP} \stackrel{?}{=} \mathbf{P}$" and their relationship to other problems. But how about questions like "$\mathbf{NP} \stackrel{?}{=} \mathbf{RP}$"?

Recall the Unambiguous Satisfiability problem

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

At first glance, you might imagine that USAT is easier than SAT itself. Imagine you're given a YES (satisfiable) instance φ, and you're trying to find a satisfying assignment. If you're promised that φ has at most one satisfying assignment, you might imagine being able to use this information to guide your search.

The Valiant-Vazirani Theorem shows that this is <u>not</u> the case, and that (at least for randomized algorithms), Unambiguous SAT is at least as hard as SAT itself.

Theorem 9 (Valiant-Vazirani)**.** *There exists a poly-time randomized algorithm A mapping n-variable CNF formulas to n-variable CNF formulas such that*

$$\begin{aligned}\varphi \in \mathsf{SAT} &\implies \Pr[A(\varphi) \in \mathsf{USAT}_Y] \geq \frac{1}{8n}\\ \varphi \notin \mathsf{SAT} &\implies \Pr[A(\varphi) \in \mathsf{USAT}_N] = 1.\end{aligned}$$

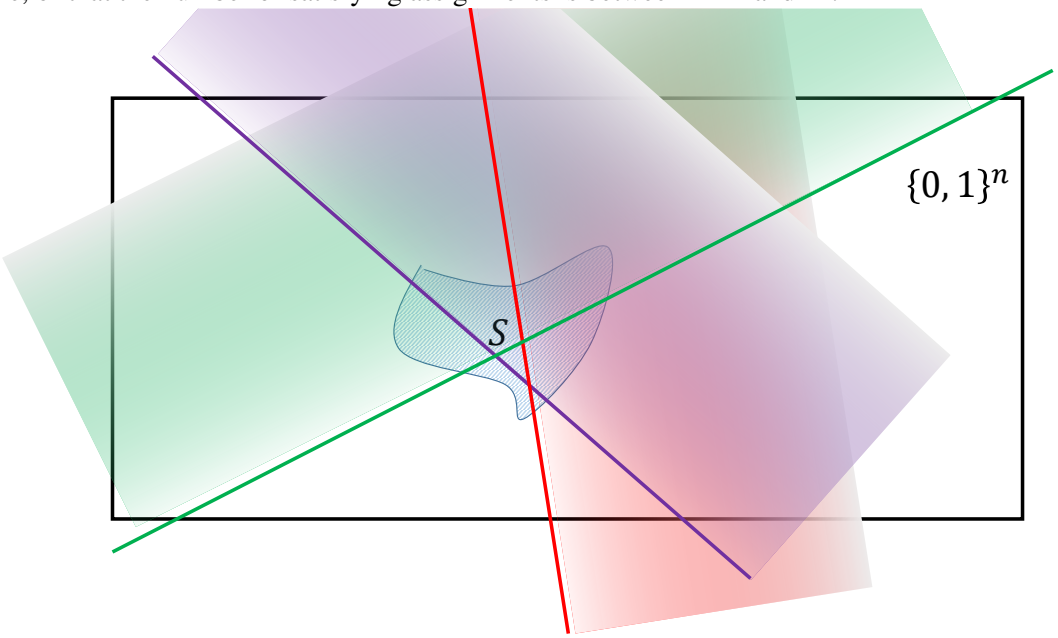
This result has a number of interesting consequences. (Left as exercises.) One is the headline statement about the hardness of USAT:

Corollary 10. $\mathbf{NP} = \mathbf{RP}$ *if and only if* $\mathsf{USAT} \in \mathbf{PrRP}$.

Another is an application to search-to-decision for SAT. Recall that the search-to-decision reduction we saw builds a satisfying assignment to a formula φ by setting one variable at a time, solving the SAT decision problem on an adaptively chosen sequence of inputs. If the satisfying assignment is unique, then these inputs can be chosen *non-adaptively*, all in a single batch; hence, Valiant-Vazirani gives a non-adaptive randomized search-to-decision reduction for SAT.

Proof idea. The reduction A will add randomly chosen additional constraints to φ to eliminate all but 1 satisfying assignment (if one exists).

In pictures. Suppose we have some additional information about our input formula φ: Either it's unsatisfiable, or that the number of satisfying assignments is between 2^{k-1} and 2^k.



Let S be the set of satisfying assignments to φ.

We'll add new constraints $C_1, \ldots, C_k$ which each (almost) independently eliminate a random half of $\{0,1\}^n$ as possible satisfying assignments. The expected size of the portion of S that survives is $2^{-k} \cdot |S| \in [1/2, 1]$. We'll show that $|S| = 1$ with constant probability.

One remaining issue: How do we know how to set k? We don't know! We'll just guess it at random from $1, \ldots, n$, which will be correct with probability $1/n$.