Computability and Complexity

Lecture 11, Tuesday September 5, 2023

Ari Feiglin

11.1 Probabilistic Algorithms

Definition 11.1:

A **probabilistic algorithm** (Turing machine) is a non-deterministic algorithm (respectively, Turing machine) such that whenever there are multiple choices for a step, each choice is chosen with uniform probability. So the output of a probabilistic algorithm is a random variable, in the case of an algorithm which solves a decision problem will take on boolean values.

Example 11.2:

Let us define the decision problem PolyIdTesting where given two multivariate polynomials, it determines if they are identically equal. The input for the problem are two polynomials, which are represented as an algebraic formula which can contain addition, multiplication, and parentheses.

The naive solution here is to expand out the algebraic formula and add the terms (of the form $c \cdot x_1^{k_1} \cdots x_n^{k_n}$) together, and compare the results for both polynomials. But this is exponential, if there are n variables then let us look at $(x_1 + \cdots + x_n)^d$ then this will end up having $\binom{n+d}{d}$ terms (the number of terms is equal to the number of ways to have $k_1 + \cdots + k_n = d$, and this is equal to the number of ways to order n dividers with d units). If we take $d = \frac{n}{2}$ this becomes exponential, so this naive solution will not solve the problem in exponential time.

Our probabilistic algorithm will function as follows: first it finds the maximum degree of a term in both p and q, let it be d (this can be done deterministically). Then if there are n variables, it chooses n values from $S = \{1, \ldots, 2d\}$. It then evaluates the polynomials at these values, and if the results are equal it returns one, otherwise it returns zero. This algorithm runs in polynomial time.

Note that this algorithm will always return the correct answer when the correct answer is "yes" (no false negatives). If the polynomials are not identical, the algorithm can be incorrect (false positive). The Schwartz-Zippel lemma states that if values $r_1, \ldots, r_n$ are chosen uniformly from a set S then

$$\mathbb{P}(p(r_1, \ldots, r_n) = q(r_1, \ldots, r_n) \mid p \neq q) \leq \frac{d}{|S|}$$

and since $|S| = 2d$ here, the algorithm will give a false positive with a probability of $\leq \frac{1}{2}$.

Notice that this algorithm does not decide PolyIdTesting, but we will discuss soon what we care about with regards to decision making with probabilistic algorithms. This algorithm does show that PolyIdTesting $\in$ **coNP**, but that's not what we're focusing on right now.

Proposition 11.3:

Let S be a decision problem and M be a polynomial-time probabilistic Turing machine such that for every x,

$$\mathbb{P}(M(x) \text{ returns the correct answer}) = 1$$

Then there exists a polynomial-time deterministic Turing machine M' which solves S.

Proof:

We define M' such to be like M, but we remove every choice. Meaning if there are multiple possible transitions from a state in M, we will define M' to only have one of the transitions. Equivalently, we could say that M' always chooses the first choice. Suppose that M' does not decide S, so there is some string x for which $M'(x)$ does not return the correct answer. But then there is a sequence of choices which M could make to return the wrong answer for x, meaning

the probability that $M(x)$ returns the correct answer is less than one, in contradiction. ■

Definition 11.4:

We define the class **RP** (randomized polynomial time) to be the set of decision problems S such that there exists a polynomial-time probabilistic algorithm M where if

$$x \in S \implies \mathbb{P}(M(x) = 1) \geq \frac{1}{2}$$
$$x \notin S \implies \mathbb{P}(M(x) = 0) = 1$$

Notice then that S is in **coRP** if and only if there exists a probabilistic algorithm M (which is the inversion of probabilistic algorithm for S^c) where if

$$x \in S \implies \mathbb{P}(M(x) = 1) = 1$$
$$x \notin S \implies \mathbb{P}(M(x) = 0) \geq \frac{1}{2}$$

So PolyIdTesting $\in$ **coRP**.

If $S \in \mathbf{RP}$ and M satisfies the definition, then M satisfies the requirements for S to be in **NP**. This is because if $x \in S$ then there exists a run of M where $M(x) = 1$ (since the probability is non-zero), and if $x \notin S$ then since the probability that $M(x) = 0$ is one, $M(x)$ will always return zero. This is precisely the requirement for S to be in **NP**. Also trivially $\mathbf{P} \subseteq \mathbf{RP}$, so we have shown

Proposition 11.5:

$$\mathbf{P} \subseteq \mathbf{RP} \subseteq \mathbf{NP}$$

Definition 11.6:

We define the class **BPP** (bounded-error probabilistic polynomial time) to be the class of decision problems S where there exists a probabilistic algorithm M such that for every x,

$$\mathbb{P}(M(x) \text{ gives the correct answer}) \geq \frac{2}{3}$$

Proposition 11.7:

$$\mathbf{RP} \subseteq \mathbf{BPP}$$

Proof:

Suppose $S \in \mathbf{RP}$, and M is a probabilistic algorithm which satisfies the requirements for S to be in **RP**. Let x be a string, let us define $M'(x)$ which runs $M(x)$ twice and return one if either run returns one. If $x \in S$ then the probability $M(x)$ returns zero both times (meaning $M'(x)$ returns the wrong answer) is $\leq \frac{1}{4}$. So if $x \in S$ the probability $M'(x)$ returns the correct answer is $\geq \frac{3}{4} \geq \frac{2}{3}$. And if $x \notin S$ then $M'(x)$ will always return the correct answer. Therefore $M'(x)$ will always return the correct answer with a probability of at least $\frac{2}{3}$, meaning $S \in \mathbf{BPP}$. ■

Notice that **coBPP** = **BPP**. This is because if $S^c \in \mathbf{BPP}$ then suppose M satisfies the condition for S^c to be in **BPP**. If we define $M' = 1 - M$, then

$$\mathbb{P}(M'(x) = (x \in S)) = \mathbb{P}(1 - M(x) = (x \in S)) = \mathbb{P}(M(x) = x \notin S) = \mathbb{P}(M(x) = x \in S^c) \geq \frac{2}{3}$$

(The notation $(x \in S)$ is the boolean value of the formula, it is 1 if x is in S and zero otherwise.) Thus by the above proposition, **coRP** $\subseteq$ **BPP** as well.

The strategy used in the proof of the proposition above, of running a probabilistic algorithm multiple times in order to improve the probability of success, is called *amplification*.

Definition 11.8:

Let $\alpha\colon \mathbb{N} \longrightarrow [0,1]$ be a function. We define the class $\mathbf{RP}_\alpha$ similar to $\mathbf{RP}$, where S is in $\mathbf{RP}$ if and only if there exists a probabilistic algorithm M such that

$$x \in S \implies \mathbb{P}(M(x)=1) \geq \alpha(|x|)$$
$$x \notin S \implies \mathbb{P}(M(x)=0) = 1$$

Notice that $\mathbf{RP} = \mathbf{RP}_{\frac{1}{2}}$. And that for every α, $\mathbf{RP}_\alpha \subseteq \mathbf{NP}$. And $\mathbf{NP} = \mathbf{RP}_{\frac{1}{2^{p(n)}}}$ (meaning the union of the classes over $p(n)$ polynomial), since $\mathbf{NP}$ makes an exponential number of guesses.

Proposition 11.9:

Let p and q be two polynomials (bound below by 2), then

$$\mathbf{RP}_{\frac{1}{p(n)}} = \mathbf{RP} = \mathbf{RP}_{1-\frac{1}{2^{q(n)}}}$$

This bounds the probability of error for $\mathbf{RP}$ from below and from above. Meaning the error can get polynomially close to 0 and exponentially close to 1.

Proof:

Obviously if $\alpha(n) \leq \beta(n)$ then $\mathbf{RP}_\beta \subseteq \mathbf{RP}_\alpha$. So we have

$$\mathbf{RP}_{\frac{1}{p(n)}} \supseteq \mathbf{RP} \supseteq \mathbf{RP}_{1-\frac{1}{2^{q(n)}}}$$

So we will show that

$$\mathbf{RP}_{\frac{1}{p(n)}} \subseteq \mathbf{RP}_{1-\frac{1}{2^{q(n)}}}$$

and then we will have finished. Let $S \in \mathbf{RP}_{\frac{1}{p(n)}}$ so there exists a probabilistic algorithm M such that when $x \in S$, $\mathbb{P}(M(x)=1) \geq \frac{1}{p(n)}$. Suppose we run $M(x)$ for $f(n)$ times, and we return one if at least one of these times returns one.

Meaning we define

1. **function** $M'(x)$
2. **repeat** $f(|x|)$ **times**
3. **if** $(M(x)=1)$ **return** 1
4. **end repeat**
5. **return** 0
6. **end function**

If $x \notin S$, then $M(x)=0$ no matter what so $M'(x)=0$. If $x \in S$, then the probability $M'(x)$ returns the wrong answer (0) is if $M(x)=0$ every time. This has a probability of

$$\leq \left(1 - \frac{1}{p(n)}\right)^{f(n)}$$

So

$$\mathbb{P}(M'(x)=1) \geq 1 - \left(1 - \frac{1}{p(n)}\right)^{f(n)}$$

we want this to be greater than $1 - \frac{1}{2^{q(n)}}$ and so we need

$$\left(1 - \frac{1}{p(n)}\right)^{f(n)} \leq 2^{-q(n)}$$

Now, notice that

$$\left(1 - \frac{1}{p(n)}\right)^{p(n)} \leq e^{-1} \leq 2^{-1}$$

and so

$$\left(1 - \frac{1}{p(n)}\right)^{p(n)\cdot q(n)} \leq 2^{-q(n)}$$

So let us choose $f(n) = p(n) \cdot q(n)$. Now, since $f(n)$ is polynomial in n, M''s runtime is polynomial in n (it runs M, which is polynomial-time, a polynomial number of times). And so M' satisfies the conditions for $S \in \mathbf{RP}_{1-\frac{1}{2^{q(n)}}}$, as required. ■

Definition 11.10:

For a function $\alpha\colon [0,1] \longrightarrow \mathbb{N}$ we define the class $\mathbf{BPP}_\alpha$ of decision problems S where there exists a polynomial-time probabilistic algorithm M such that the probability $M(x)$ returns a correct answer is greater than $\alpha(|x|)$.

Theorem 11.11:

Let p, q are polynomials (greater than 2), then

$$\mathbf{BPP}_{\frac{1}{2}+\frac{1}{q}} = \mathbf{BPP} = \mathbf{BPP}_{1-2^{-p}}$$

Proof:

Obviously again if $\alpha \leq \beta$ then $\mathbf{BPP}_\beta \subseteq \mathbf{BPP}_\alpha$. We will show the second equality, so all we need to do is show $\mathbf{BPP} \subseteq \mathbf{BPP}_{1-2^{-p}}$. If $S \in \mathbf{BPP}$, let M satisfy the conditions for S to be in $\mathbf{BPP}$. We define

1. **function** $M'(x)$
2. $\quad c_0, c_1 \leftarrow 0$
3. $\quad$ **repeat** $k(|x|)$ **times**
4. $\quad\quad$ **if** $(M(x) = 1)$ $\; c_1 \leftarrow c_1 + 1$
5. $\quad\quad$ **else** $c_0 \leftarrow c_0 + 1$
6. $\quad$ **end repeat**
7. $\quad$ **return** 1 **if** $c_1 > c_0$, **else** 0
8. **end function**

Let us define the indicator ω_i which indicates if M was incorrect on the ith iteration. Thus

$$\mathbb{E}[\omega_i] = \mathbb{P}(M(x) \text{ is incorrect}) \leq \frac{1}{3}$$

And so

$$\mathbb{E}\left[\frac{\sum_{i=1}^k \omega_i}{k}\right] = \frac{1}{k}\sum_{i=1}^k \mathbb{E}[\omega_i] \leq \frac{k}{3k} = \frac{1}{3}$$

This means that we expect at most $\frac{1}{3}$ of the iterations will give the incorrect result. And we know that $M'(x)$ is incorrect when most of the results are incorrect, ie. $M'(x)$ is incorrect when

$$\frac{\sum_{i=1}^k \omega_i}{k} > \frac{1}{2}$$

Recall Chernoff's inequality: if $\omega_1, \ldots, \omega_k$ are random variables with the same distributions bounded by $a < b$, then

$$\mathbb{P}\left(\frac{1}{k}\sum_{i=1}^k \omega_i > \mu + \varepsilon\right) \leq e^{-2\varepsilon^2 k/(b-a)^2}, \qquad \mu = \mathbb{E}\left[\frac{1}{k}\sum_{i=1}^k \omega_i\right]$$

So from Chernoff's inequality, we get that (the bounds are $a = 0$ and $b = 1$)

$$\mathbb{P}(M(x) \text{ is incorrect}) = \mathbb{P}\left(\frac{1}{k}\sum_{i=1}^k \omega_i > \frac{1}{2} = \mu + \frac{1}{6}\right) \leq e^{-k/18}$$

So let $k(n) = 18p(n)$, then we get that

$$\mathbb{P}(M(x) \text{ is incorrect}) \leq e^{-p(n)} \leq 2^{-p(n)}$$

and so $M(x)$ is correct with probability greater than $1 - 2^{-p(n)}$.

Since M' runs M, which is polynomial-time, a polynomial number of times ($k(n)$ is polynomial), M' is polynomial-time. And we showed that it is correct with probability greater than $1 - 2^{-p}$, meaning $S \in \mathbf{BPP}_{1-2^{-p}}$ as required.

The first equality is proven in a similar matter. ■

Proposition 11.12:
BPP is closed under unions.

Proof:

Suppose S_1 and S_2 are in **BPP**. By above, there exist probabilistic algorithms M_1 and M_2, for S_1 and S_2, which return the correct answer with probabilities $\geq \frac{5}{6}$. Let us define $M(x)$ which simply runs $M_1(x)$ and $M_2(x)$ and returns one if either returns one.

If $x \in S_1 \cup S_2$, without loss of generality $x \in S_1$. Then

$$\mathbb{P}(M(x) = 1) \geq \mathbb{P}(M_1(x) = 1) \geq \frac{5}{6} \geq \frac{2}{3}$$

Now if $x \notin S_1 \cup S_2$ then

$$\mathbb{P}(M(x) = 0) = \mathbb{P}(M_1(x) = M_2(x) = 0) = \mathbb{P}(M_1(x) = 0) \cdot \mathbb{P}(M_2(x) = 0) \geq \frac{25}{36} \geq \frac{2}{3}$$

So M satisfies the conditions for $S_1 \cup S_2 \in$ **BPP**. ■

Note that it is an unknown what the relation between **BPP** and **NP**.

Proposition 11.13:
If **NP** $\subseteq$ **BPP** then **NP** = **RP**.

Proof:

Firstly, in general **RP** $\subseteq$ **NP**, so all we must show is the other direction: that **NP** $\subseteq$ **RP**. We will show that $\mathsf{SAT} \in$ **RP**, and since **RP** is closed under Karp reductions (this is simple to prove) this means that **NP** $\subseteq$ **RP**. Since $\mathsf{SAT} \in$ **NP** $\subseteq$ **BPP**, there exists a probabilistic algorithm $M(\varphi)$ which returns correct answers for $\varphi \in \mathsf{SAT}$ with a probability of $\alpha(n)$. We don't know what α is yet, we will show that there exists an α which helps us solve the problem later in the proof.

Let us define

1. **function** $M'(\varphi)$
2. Let τ be a boolean vector
3. $\varphi' \leftarrow \varphi$
4. **for** (i **from** 1 **to** $|\text{Var}\varphi|$) ▷ Varφ *are the variables in* φ
5. Evaluate the first variable x as true, and simplify φ'
6. **if** ($M(\varphi') = 1$) $\tau_i \leftarrow \mathsf{true}$
7. **else** $\tau_i \leftarrow \mathsf{false}$
8. **end for**
9. Check if τ satisfies φ, and return 1 if it does, 0 otherwise.
10. **end function**

If $\varphi \notin \mathsf{SAT}$ then $M'(\varphi)$ will return zero as the boolean vector τ which it constructs will not satisfy φ. And if $\varphi \in \mathsf{SAT}$, if M returned a correct answer over every iteration then τ will indeed satisfy φ and so $M'(\varphi) = 1$. So then

$$\mathbb{P}(M'(\varphi) = 1) \geq \mathbb{P}(M(\varphi) \text{ returned the correct answer over every iteration})$$

The complement of this is (let $k = |\text{Var}\varphi|$)

$$\mathbb{P}(M(\varphi) \text{ gave an incorrect answer for some iteration}) = \mathbb{P}\left(\bigcup_{i=1}^{k} M(\varphi) \text{ gave an incorrect answer on the } i\text{th iteration}\right)$$

By the union bound, we get

$$\leq \sum_{i=1}^{k} \mathbb{P}(M(\varphi) \text{ gave an incorrect answer on the } i\text{th iteration})$$

The probability $M(\varphi)$ is false is $1 - \alpha$, so this is equal to $n(1 - \alpha)$. Thus we get that

$$\mathbb{P}(M'(\varphi) = 1) \geq 1 - n(1 - \alpha)$$

We want $\mathbb{P}(M'(\varphi) = 1) \geq \frac{1}{2}$ by the definition of **RP**, and so we require

$$1 - n(1 - \alpha) \geq \frac{1}{2}$$

And so we set

$$\alpha(n) = 1 - \frac{1}{2n}$$

And so if we can show that $\mathbf{BPP} \subseteq \mathbf{BPP}_\alpha$, we have shown that SAT is in **RP** (since then this would justify our claim that M succeeds with probability α). But this is true since $1 - 2^{-n} \geq 1 - \frac{1}{2n}$ and we showed in a theorem above that for αs of the form $1 - 2^{-p}$, $\mathbf{BPP}_\alpha = \mathbf{BPP}$. ■