Lecture 2

Relation of Polynomial-time Hierarchy, Circuits, and Randomized Computation

April 3, 2008
Lecturer: Paul Beame
Notes:

2.1 Turing Machines with Advice

Last lecture introduced non-uniformity through circuits. An alternate view of non-uniformity is Turing machines with an advice tape. The advice tape contains some extra information that depends only on the length of the input; i.e., on input x the TM gets $(x, \alpha_{|x|})$.

Definition 2.1 $\mathsf{TIME}(t(n))/f(n) = \{A \mid A$ is decided in time $O(t(n))$ by a TM with advice sequence $\{\alpha_n\}_n$ such that $\alpha_n \in \{0,1\}^{f(n)}\}$. (Note that the we ignore constant factors in the running time but not the advice.)

Now we can define the class of languages decidable in polynomial time with polynomial advice:

Definition 2.2 $\mathsf{P/poly} = \bigcup_{k,\ell} \mathsf{TIME}(n^k)/n^\ell$

Lemma 2.3 $\mathsf{P/poly} = \mathsf{POLYSIZE}$.

Proof $\mathsf{POLYSIZE} \subseteq \mathsf{P/poly}$: Given a polynomial-size circuit family $\{C_n\}_n$ produce a $\mathsf{P/poly}$ TM M by using advice strings $\alpha_n = \langle C_n \rangle$. On input x, M can then evaluate circuit $C_{|x|}$ on input x in time polynomial in $|x|$ and $|\langle C_{|x|}\rangle|$ which is polynomial in $|x|$.

$\mathsf{P/poly} \subseteq \mathsf{POLYSIZE}$: Given a $\mathsf{P/poly}$ TM M with advice strings $\{\alpha_n\}_n$, use the tableau construction from the Cook-Levin Theorem to construct a polynomial size circuit family with the advice strings hard-coded in the circuit. $\square$

If $\mathsf{NP} \subseteq \mathsf{P}$ then $\mathsf{PH} = \mathsf{P}$ but although $\mathsf{P/poly}$ contains undecidable languages we still get a collapse of the polynomial-time hierarchy if $\mathsf{NP} \subseteq \mathsf{P/poly}$.

Theorem 2.4 (Karp-Lipton) If $\mathsf{NP} \subseteq \mathsf{P/poly}$ then $\mathsf{PH} = \Sigma_2^p \cap \Pi_2^p$.

Proof Assume that $\mathsf{NP} \subseteq \mathsf{P/n^{O(1)}}$. It suffices to show that this implies that $\Pi_2^p \subseteq \Sigma_2^p$. In particular we use the fact there any NP problem has a polynomial-time circuit to find a Σ_2^p algorithm for the Π_2^p-complete problem $\Pi_2 SAT$. Recall that $\langle\varphi\rangle \in \Pi_2 SAT$ if and only if

$$\forall u \in \{0,1\}^n \exists v \in \{0,1\}^n \, \varphi(u,v).$$

Observe that $\{(\langle\varphi\rangle, u) \mid \exists v \in \{0,1\}^n\, \varphi(u,v)\}$ is an NP language. Therefore, by assumption, there exists a circuit family $\{C_n\}_n$ of size $q(n)$ for some polynomial q such that $C_n(\langle\varphi\rangle, u) = 1$ if and only if $\exists v \in \{0,1\}^n\, \varphi(u,v)$. It would be then seem natural to define a Σ_2^p algorithm to existentially quantify over the bits of the encoding of $\langle C_n\rangle$ and then universally quantify over u. The difficulty is that we don't know that the bits sequence actually is for the correct circuit C_n that actually solve the NP problem. However, by applying the standard polynomial self-reduction for NP problems we can convert the circuit family $\{C_n\}_n$ to a circuit family $\{C'_n\}_n$ that on input $(\langle\varphi\rangle, u)$ actually produces a $v' \in \{0,1\}^n$ such that $\varphi(u, v')$ is true if one exists. (The circuit C'_n will have to make n calls to the circuit C_n successively fixing one bit of v' at a time so its size will be at most $nq(n)$ and thus $\langle C'_n\rangle$ will be at most $n^2q^2(n)$ bits long.) Therefore the Σ_2^p characterization of $\Pi_2 SAT$ is

$$\exists\langle C'_n\rangle \forall u \in \{0,1\}^n, \varphi(u, C'_n(\langle\varphi\rangle, u)),$$

which is what we needed to show □

2.2 Probabilistic Complexity Classes

A *probabilistic (randomized) TM* is an ordinary multi-tape TM with an extra one-way read-only *coin flip (random)* tape. If the running time of M is $T(n)$ then on input x, the coin flip tape is initialized to a uniformly random string $r \in \{0,1\}^{f(|x|)}$ where $f(n) \leq T(n)$. If r is the string of coin flips for a machine M then we write $T(n)$ then $|r| \leq T(n)$. Now we can write $M(x,r)$ to denote the output of M on input x with random tape r where $M(x,r) = 1$ if M accepts and $M(x,r) = 0$ if M rejects.

A *probabilistic polynomial-time Turing Machine (PPT)* is a probabilistic Turing Machine whose worst-case running time $T(n)$ is polynomial in n.

We can now define several probabilistic complexity classes. (The terminology, due to Gill who introduced these classes, is not the most natural but it has stuck.)

Definition 2.5 Randomized Polynomial Time: $L \in \mathsf{RP}$ if and only if there exists a probabilistic polynomial time TM M such that for some error $\epsilon < 1$,

- $\forall w \in L$, $\Pr[M \text{ accepts } w] \geq 1 - \epsilon$, and
- $\forall w \notin L$, $\Pr[M \text{ accepts } w] = 0$.

equivalently $\forall w \in L$, $\Pr_r[M(w,r) = 1] \geq 1 - \epsilon$ and $\forall w \notin L$, $\Pr_r[M(w,r) = 1] = 0$.

The error, ϵ, is fixed for all input sizes. RP is the class of problems with one-sided error (i.e. an accept answer is always correct, whereas a reject may be incorrect.) coRP, which has one-sided error in the other direction, is defined analogously. The following class encompasses machines with two-sided error:

Definition 2.6 Bounded-error Probabilistic Polytime: $L \in \mathsf{BPP}$ if and only if there exists a probabilistic polynomial time TM M such that for some $\epsilon < \frac{1}{2}$,

- $\forall w \in L$, $\Pr[M \text{ accepts } w] \geq 1 - \epsilon$ and
- $\forall w \notin L$, $\Pr[M \text{ accepts } w] \leq \epsilon$.

If we identify the language L with its characteristic function $L(w) = \begin{cases} 1 & \text{if } w \in L \\ 0 & \text{if } w \notin L \end{cases}$ then we can write this equivalently as $L \in \mathsf{BPP}$ iff for all w we have $\Pr_r[M(w,r) = L(w)] \geq 1 - \epsilon$.

Clearly $\mathsf{RP} \subseteq \mathsf{NP}$, $\mathsf{coRP} \subseteq \mathsf{coNP}$. Also $\mathsf{RP}, \mathsf{coRP} \subseteq \mathsf{BPP}$ and BPP is closed under complement. Randomized algorithms with 1-sided or 2-sided errors such as these are known as *Monte Carlo* algorithms. Although we have so far required that the error in the definitions of BPP,RP, and coRP be constant we can consider the more general case when the error $\epsilon = \epsilon(n)$ is a function of the input size.

Definition 2.7 Zero-error Probabilistic Polytime: $\mathsf{ZPP} = \mathsf{RP} \cap \mathsf{coRP}$.

Lemma 2.8 $L \in \mathsf{ZPP}$ if and only if there is a probabilistic TM M that always outputs the correct answer (i.e, $L(M) = L$) and the expected runtime of M is polynomial.

Proof
$\Rightarrow$: Let M_1 be an RP machine for L, and M_2 be a coRP machine for L with errors $\epsilon_1, \epsilon_2 < 1$. Define a probabilistic TN M that repeatedly runs M_1 followed by M_2 using independent random strings until one accepts. If either accepts then the answer must be correct so if M_1 accepts, then accept and if M_2 accepts then reject. Let $\epsilon = \max(\epsilon_1, \epsilon_2)$. We expect to have to run at most $\frac{1}{1-\epsilon}$ trials before one accepts. Thus M decides L in polynomial expected time.

$\Leftarrow$: Let $T(n)$ be the expected running time of a probabilistic TM M that always outputs the correct answer for language L. By Markov's inequality the probability that M runs for more than $3T(n)$ steps is at most $1/3$. To get an RP algorithm M' for L truncate the computation of M after $3T(n)$ steps. If M has accepted then accept, otherwise reject. If $w \in L$ then M' will accept w with probability at least $2/3$ and if $w \notin L$ then M will not accept w no matter what the random string. The algorithm for coRP is completely dual. $\square$

Randomized algorithms that are always correct but may run forever are known as Las Vegas algorithms. Our last probabilistic complexity class is much more powerful:

Definition 2.9 Probabilistic Polytime: $L \in \mathsf{PP}$ if and only if

$$\Pr_r[M(w,r) = L(w)] > \frac{1}{2}.$$

Here the error is allowed to be exponentially close to $1/2$, which is the key difference from BPP.

Note that with PP, it might take exponentially many trials even to notice the probability advantage.

2.2.1 Amplification

Lemma 2.10 For any probabilistic TM M with running time $T(n)$ and two-sided error $\epsilon(n) = \frac{1}{2} - \delta(n)$ there is a probabilistic TM M' with running time at most $O(\frac{m}{\delta^2(n)}T(n))$ and error at most 2^{-m} for the same language.

Proof M' simply runs M some number, k, times and takes the majority vote. The result follows by simple Chernoff bounds. We give a detailed calculation below. The error is:

$$\begin{aligned}
\Pr_r[M'(x,r) \neq L(x)] &= \Pr[\geq \frac{k}{2} \text{ wrong answers on } x] \\
&= \sum_{i=0}^{k/2} Pr[\frac{k}{2} + i \text{ wrong answers of } M \text{ on } x] \\
&= \sum_{i=0}^{k/2} \binom{k}{\frac{k}{2}+i} \epsilon^{\frac{k}{2}+i} (1-\epsilon)^{\frac{k}{2}+i} \\
&\leq \sum_{i=0}^{k/2} \binom{k}{\frac{k}{2}+i} \epsilon^{\frac{k}{2}} (1-\epsilon)^{\frac{k}{2}} \\
&\leq 2^k \epsilon^{\frac{k}{2}} (1-\epsilon)^{\frac{k}{2}} \\
&= \left[4(\frac{1}{2}-\delta)(\frac{1}{2}+\delta)\right]^{\frac{k}{2}} \\
&= (1-4\delta^2)^{\frac{k}{2}} \\
&\leq e^{-2\delta^2 k} \quad \text{since } 1-x \leq e^{-x} \\
&\leq 2^{-m} \quad \text{for } k = \frac{m}{\delta^2}
\end{aligned}$$

□

Note that amplification from sub-constant to constant error allows us to generalize the definition of BPP to allow $\epsilon = \epsilon(n) = 1/2 - \delta(n)$ for $\delta(n) \geq 1/q(n)$ for any polynomial q. However for PP, this amplification does not yield an efficient algorithm since $\delta(n)$ may be 2^{-n}.

A similar approach can be used with an RP language, this time accepting if any of the k trials accept. This gives an error of ϵ^k, where we can choose $k = \frac{m}{\log(\frac{1}{\epsilon})}$.

2.3 Randomness and Non-uniformity

The following theorem show that randomness is no more powerful than advice in general.

Theorem 2.11 (Gill, Adleman) BPP $\subseteq$ P/poly.

Proof Let $L \in$ BPP. By the amplification lemma, there exists a BPP machine M for L and a polynomial p such that:

$$\forall x \Pr_{r \in \{0,1\}^{p(n)}} [M(x,r) \neq L(x)] \leq 2^{-n-1}.$$

For $r \in \{0,1\}^{p(n)}$ say that r is *bad for x* iff $M(x,r) \neq L(x)$. By assumption, for all $x \in \{0,1\}^n$,

$$\Pr_r[r \text{ is bad for } x] \leq 2^{-n-1}$$

We say that r is *bad* if there exists an $x \in \{0,1\}^n$ such that r is bad for x.

$$\begin{aligned}\Pr_r[r \text{ is bad}] &\leq \sum_{x\in\{0,1\}^n} \Pr_r[r \text{ is bad for } x] \\ &\leq 2^n 2^{-n-1} \leq 1/2 < 1.\end{aligned}$$

Therefore for every n there must exist an $r_n \in \{0,1\}^{p(n)}$ such that r_n is not bad. (In fact this is true by construction for at least half the strings in $\{0,1\}^{p(n)}$.) We can use this sequence $\{r_n\}_n$ as the advice sequence for a $\mathsf{P/poly}$ machine that decides L. Each advice string is a particular random string r_n that leads to a correct answer for every input of length n. $\square$

2.4 BPP and the Polynomial-time Hierarchy

We know that $\mathsf{RP} \subseteq \mathsf{NP}$. Here we see that generalizing to bounded 2-sided error still stays within PH.

Theorem 2.12 (Sipser-Gacs, Lautemann) $\mathsf{BPP} \subseteq \Sigma_2^p \cap \Pi_2^p$

Proof Note that BPP is closed under complement, so it suffices to show $\mathsf{BPP} \subseteq \Sigma_2^p$.
Let $L \in \mathsf{BPP}$. Then by amplification, there is a probabilistic polytime TM M and polynomial $p(n)$ such that

$$\Pr_{r\in\{0,1\}^{p(n)}}[M(x,r) \neq L(x)] \leq 2^{-n}.$$

Define $Acc_M(x) = \{r \in \{0,1\}^{p(n)} \mid M(x,r) = 1\}$. We have two cases: either $Acc_M(x)$ is almost all of $\{0,1\}^{p(n)}$ and we should accept x or $Acc_M(x)$ is only an exponentially small fraction of $\{0,1\}^{p(n)}$ and we should reject x. Moreover, we have a polynomial-time algorithm to determine membership in $Acc_M(x)$, namely on input r simply run $M(x,r)$.

The general property we will prove is that for if a set $S \subseteq \{0,1\}^m$ contains a large fraction of $\{0,1\}^m$ then a small number of translations of S will cover $\{0,1\}^m$ but if S is a small fraction of $\{0,1\}^m$ then no small set of translations will suffice to cover the set. The translation we use is just bit-wise exclusive or of bit vectors, $\oplus$. For $S \subseteq \{0,1\}^m$ and $t \in \{0,1\}^m$, define $S\oplus t = \{s\oplus t | s \in S\}$. Note that $|S \oplus t| = |S|$ and that $b \in S \oplus t$ if and only if $t \in S \oplus b$.

Lemma 2.13 (Lautemann) Let $S \subseteq \{0,1\}^m$. If $\frac{|S|}{2^m} > \frac{1}{2}$ then there exists $t_1, \ldots, t_m \in \{0,1\}^m$ such that,

$$\bigcup_{j=1}^{m}(S \oplus t_j) = \{0,1\}^m.$$

Proof By the probabilistic method.
Let $|S| > 2^{m-1}$ be a sufficiently large set as defined above. Choose $t_1, \cdots, t_m$ uniformly and independently at random from $\{0,1\}^m$. Fix a string $b \in \{0,1\}^m$ and $j \in [m] = \{1, \ldots, m\}$.

$$\Pr[b \in S \oplus t_j] = \Pr[t_j \in S \oplus b] = \Pr[t_j \in S] > \frac{1}{2}.$$

Therefore for any $j \in [m]$, $\Pr[b \notin S \oplus t_j] < 1/2$. The probability that b is not in any of the m translations is then

$$\Pr[b \notin \bigcup_{j=1}^{m}(S \oplus t_j)] = \prod_{j=1}^{m} \Pr[b \notin S \oplus t_j] < 2^{-m}.$$

Therefore

$$\Pr[\exists b \in \{0,1\}^m \text{ s.t. } b \notin \bigcup_{j=1}^{m}(S \oplus t_j)] < 2^m 2^{-m} = 1.$$

Therefore there exists a set $t_1, \ldots, t_m$ such that the union of the translations of S by the t_j covers all strings in $\{0,1\}^m$. □

Now apply Lautemann's lemma with $S = Acc_M(x)$ and $m = p(n)$. If $x \notin L$ then $Acc_M(x)$ is only a 2^{-n} fraction of $\{0,1\}^m$, and so m translations will only be able to cover at most an $p(n)2^{-n}$ fraction of $\{0,1\}^m$, certainly not all of it. This gives us the following Σ_2^p characterization of L:

$$x \in L \Leftrightarrow \exists(t_1, \ldots, t_{p(|x|)}) \in \{0,1\}^{p^2(|x|)} \forall r \in \{0,1\}^{p(|x|)} (M(x, r \oplus t_1) = 1 \vee \ldots \vee M(x, r \oplus t_{p(|x|)}) = 1).$$

□