Lecture 3

Randomized Computation and the Polynomial-time Hierarchy

April 4, 2004
Lecturer: Paul Beame
Notes: Ethan Phelps-Goodman

3.1 Undecidable Languages in POLYSIZE

By the proof of the Cook-Levin theorem we know that all languages in P have polynomial size circuits. The converse is not true, however. In fact, there are languages with polynomial size circuits that are not decidable. For example,

$$A = \{1^{\langle M,x\rangle} \mid \text{Turing machine } M \text{ halts on input } x\} \in \mathsf{SIZE}(n)$$

where $1^{\langle M,x\rangle}$ denotes the unary encoding of the binary representation of machine M and input x. The construction of the circuit is as follows: Each input size corresponds to a particular machine and input. If the machine would halt on this input, then the circuit consists of the AND of all input variables. If the machine does not halt on the input then the circuit's output gate is always 0. This seemingly strange situation arises from the way we have defined circuit families. In particular, you are allowed to use an unbounded amount of computation to construct each particular circuit. This property is called non-uniformity.

(In a uniform circuit family each circuit can be built (or the structure of the gates in its underlying graph can be decided) in some small time-bounded or space-bounded class. There are different notions of uniform circuit classes depending precisely on what notion is used. Common examples used are log-space, or polynomial time.)

In fact, even constant-size circuits can decide undecidable languages: For example,

$$A = \{x \in \{0,1\}^* \mid \text{the } |x|^{th} \text{ TM } M_{|x|} \text{ halts on input } 1^{|x|}\} \in \mathsf{SIZE}(1).$$

3.2 Turing Machines with Advice

Last lecture introduced non-uniformity through circuits. An alternate view of non-uniformity uses 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 3.1. $\mathsf{TIME}(T(n))/f(n) = \{A \mid A$ is decided in time $O(T(n))$ by a TM with advice sequence $\{\alpha_n\}_{n=0}^{\infty}$ such that $|\alpha_n|$ is $O(f(n))\}$.

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

Definition 3.2. $\mathsf{P/poly} = \bigcup_{k,l} \mathsf{TIME}(n^k)/n^l$

Lemma 3.1. $\mathsf{P/poly}$= $\mathsf{POLYSIZE}$

Proof.
$\mathsf{POLYSIZE} \subseteq \mathsf{P/poly}$: Given a circuit family in $\mathsf{POLYSIZE}$, produce a TM with advice M that interprets its advice string as the description of a circuit and evaluates that circuit on the input. Use $\{\langle c_n \rangle\}_{n=0}^{\infty}$ as the advice strings. These are polynomial size and the evaluation is polynomial time.

$\mathsf{P/poly} \subseteq \mathsf{POLYSIZE}$: Given a TM M and a sequence of advice strings $\{\alpha_n\}_{n=0}^{\infty}$, use the tableau construction from the proof of the Cook-Levin Theorem to construct a polynomial-size circuit family with the advice strings hard-coded into the input. □

Typically, people tend to use the equivalent $\mathsf{P/poly}$ rather than $\mathsf{POLYSIZE}$ to describe the complexity class since it emphasizes the fact that it is the natural non-uniform generalization of P.

3.3 Probabilistic Complexity Classes

Definition 3.3. A *probabilistic Turing machine* is an NTM where each configuration has exactly one or two legal next moves. Steps with two moves are called *coin flip* steps. By viewing each such step as the flip of a fair coin we define the *probability* of a branch b being executed is $\Pr[b] = 2^{-k}$, where k is the number of coin flip steps along branch b. Then we define the probability of acceptance as:

$$\Pr[M \text{ accepts } w] = \sum_{\text{branches } b \text{ on which } M \text{ accepts } w} \Pr[b]$$

and $\Pr[M \text{ rejects } w] = 1 - \Pr[M \text{ accepts } w]$.

Probabilistic TMs can also be viewed as ordinary multi-tape TMs with an extra read-only coin flip tape. If r is the string of coin flips and machine M runs in time $T(n)$ then $|r| \leq T(n)$. Now we can write the answer of M on x as a function $M(x, r)$ which equals 1 if M accepts x given random string r.

We can now define probabilistic complexity classes.

Definition 3.4. <u>R</u>andomized <u>P</u>olynomial Time: A language $A \in \mathsf{RP}$ iff there exists a probabilistic polynomial time TM M such that for some $\epsilon < 1$,

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

The error, ϵ, is fixed for all input size. $\mathsf{RP}$is the class of problems with one-sided error (ie. an accept answer is always correct, whereas a reject may be incorrect.) We can also define coRP, which has one-sided error in the other direction. The following class encompasses machines with two-sided error:

Definition 3.5. <u>B</u>ounded-error <u>P</u>robabilistic <u>P</u>olynomial Time: A language $A \in \mathsf{BPP}$ iff there exists a probabilistic TM M running in polynomial time such that for some $\epsilon < 1/2$:

1. $\forall w \in A$, $\Pr[M$ accepts $w] \geq 1 - \epsilon$
2. $\forall w \notin A$, $\Pr[M$ rejects $w] \geq 1 - \epsilon$

We will slightly abuse notation and conflate languages and their characteristic functions; i.e., for a language A,

$$A(w) = \begin{cases} 1 & \text{if } w \in A \\ 0 & \text{if } w \notin A. \end{cases}$$

Using this we can say that $A \in \mathsf{BPP}$ iff there is some probabilistic polynomial-time TM M such that

$$\Pr[M(w, r) = A(w)] \geq 1 - \epsilon.$$

We will also define a zero-sided error complexity class:

Definition 3.6. Zero-error Probabilistic Polynomial Time: $\mathsf{ZPP} = \mathsf{RP} \cap \mathsf{coRP}$

The motivation for this terminology is the following lemma.

Lemma 3.2. *If $A \in \mathsf{ZPP}$ then there is a probabilistic TM M such that $L(M) = A$ and the expected running time of M is polynomial.*

Proof. Let M_1 be an RP machine for A, and M_2 be a coRP machine for A; i.e., an RP machine for $\overline{A}$. M repeatedly runs M_1 and M_2 alternately until one accepts. If M_1 accepts, then accept. If M_2 accepts then reject. Let $\epsilon = \max\{\epsilon_1, \epsilon_2\}$. We expect to have to run $\frac{1}{1-\epsilon}$ trials before one accepts. Thus M decides A in polynomial expected time. □

The last probabilistic complexity class is much more powerful:

Definition 3.7. Probabilistic Polytime: $A \in \mathsf{PP}$ iff there is a probabilistic polynomial time TM M such that

$$\Pr[M(w, r) = A(w)] > 1/2.$$

Here the error is allowed depend on the input size and be exponentially close to $1/2$.

Remark. Running a polynomial time experiment using a machine witnessing that $A \in \mathsf{PP}$ will in general not be enough to tell whether or not an input $x \in A$ or not because the difference between acceptance and rejection probabilities may be exponentially close to 1.

3.4 Amplifi cation

The following lemma shows that in polynomial time we can reduce errors that are polynomially close to $1/2$ to exponentially small values.

Lemma 3.3. *Let M be a probabilistic TM with two-sided error $\epsilon = 1/2 - \delta$ running in time $T(n)$. Then for any $m > 0$ there is a probabilistic polytime TM M' with runtime at most $O(\frac{m}{\delta^2} T(n))$ and error at most 2^{-m}.*

Proof. M' simply runs M some number k times on independently chosen random strings and takes the majority vote of the answers. For convenience we assume that k is even. The error is:

$$\begin{aligned}
\Pr[M'(x) \neq A(x)] &= \Pr[\geq k/2 \text{ wrong answers on } x] \\
&= \sum_{i=0}^{k/2} \Pr[k/2+i \text{ wrong answers of } M \text{ on } x] \\
&\leq \sum_{i=0}^{k/2} \binom{k}{k/2+i} \epsilon^{k/2+i}(1-\epsilon)^{k/2-i} \\
&\leq \sum_{i=0}^{k/2} \binom{k}{k/2+i} \epsilon^{k/2}(1-\epsilon)^{k/2} \qquad \text{since } \epsilon \leq 1-\epsilon \text{ for } \epsilon \leq 1/2 \\
&\leq 2^k \epsilon^{k/2}(1-\epsilon)^{k/2} \\
&= [4(1/2-\delta)(1/2+\delta)]^{k/2} \\
&= (1-4\delta^2)^{k/2} \\
&\leq e^{-2\delta^2 k} \qquad \text{since } 1-x \leq e^{-x} \\
&\leq 2^{-m} \qquad \text{for } k = m/\delta^2.
\end{aligned}$$

□

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 $k = \frac{m}{\log(\frac{1}{\epsilon})}$.

3.5 Randomness vs. Advice

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

Theorem 3.4 (Gill, Adleman). $\mathsf{BPP} \subseteq \mathsf{P/poly}$

Proof. Let $A \in \mathsf{BPP}$. By the amplification lemma, there exists a BPP machine M for A and a polynomial bound p such that for all $x \in \{0,1\}^n$,

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

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

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

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

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

Thus $\Pr[r \text{ is not bad}] > 0$. Therefore there must exist an $r_n \in \{0,1\}^{p(n)}$ such that r_n is not bad. Apply this same argument for each value of n and use this sequence $\{r_n\}_{n=0}^{\infty}$ as the advice sequence to a $\mathsf{P/poly}$ machine that decides A. Each advice string is a particular random string that leads to a correct answer for every input of that length. $\square$

3.6 BPP and the Polynomial Time Hierarchy

Here we show that randomness ican be simulated by a small amount of alternation.

Theorem 3.5 (Sipser-Gacs, Lautemann). $\mathsf{BPP} \subseteq \Sigma_2\mathsf{P} \cap \Pi_2\mathsf{P}$.

Proof. Note that BPP is closed under complement; so, it suffices to show $\mathsf{BPP} \subseteq \Sigma_2 P$.
Let $A \in \mathsf{BPP}$. Then there is a probabilistic polytime TM M and polynomial p such that for $x \in \{0,1\}^n$,

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

Define $Acc_M(x) = \{r \in \{0,1\}^{p(n)} \mid M(x,r) = 1\}$. In order to determine whether or not $x \in A$ we need to determine whether the set $S = Acc_M(x)$ is large (nearly all of $\{0,1\}^{p(n)}$) or small (only an exponentially small fraction of $\{0,1\}^{p(n)}$).

The basic idea of the method we use is the following: If a set S contains a large fraction of $\{0,1\}^{p(n}$ then a small number of "translations" of S will cover $\{0,1\}^{p(n)}$. If S is a very small fraction of $\{0,1\}^{p(n)}$ then no small set of "translations" of S will suffice to cover $\{0,1\}^{p(n)}$.

The translation we will use is just bitwise exclusive or, $\oplus$. We will use the following invertibility property of $\oplus$: for $s \in \{0,1\}^m$, $t \in \{0,1\}^m$,

$$b = s \oplus t \Longleftrightarrow t = s \oplus b$$

For $S \subseteq \{0,1\}^m$, define $S \oplus t = \{s \oplus t | s \in S\}$. Note that $|S \oplus t| = |S|$.

Lemma 3.6 (Lautemann's Lemma). *Let $S \subseteq \{0,1\}^m$. If $|S|/2^m \geq 1/2$ then there exist $t_1 \ldots t_m \in \{0,1\}^m$ such that, $\bigcup_{j=1}^m (S \oplus t_j) = \{0,1\}^m$.*

Thus the number of translations required is only linear in the number of bits in the size of the universe from which S is chosen.

Proof. By probabilistic method. Let S satisfy the conditions of the lemma.

Fix a string $b \in \{0,1\}^m$. Choose $t_1 \ldots t_m$ uniformly and independently at random from $\{0,1\}^m$. For any $j \in \{1,\ldots,m\}$,

$$\begin{aligned}\Pr[b \in S \oplus t_j] &= \Pr[t_j \in S \oplus b] \\ &= \Pr[t_j \in S] \qquad \text{since } |S \oplus t_j| = |S| \\ &\geq 1/2.\end{aligned}$$

Therefore $\Pr[b \notin S \oplus t_j] < 1/2$. Thus, by independence, the probability that b is not in any of the m translations

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

Thus

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

and so

$$\Pr[\forall b \in \{0,1\}^m.\ b \in \bigcup_{j=1}^{m} (S \oplus t_j)] > 0.$$

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

Now apply Lautemann's Lemma with $S = Acc_M(x)$ and $m = p(n)$. If $x \notin A$ then only a 2^{-n} fraction of the random strings will be in $Acc_M(x)$, and so $m = p(n)$ translations will not be able to cover all of $\{0,1\}^{p(n)}$. This gives us the following Σ_2P characterization of A:

$x \in A \iff \exists t_1 \ldots t_{p(|x|)} \in \{0,1\}^{p(|x|)} \forall^{p(|x|)} r$ there is some $j \in \{1, \ldots, p(|x|)\}$ such that $M(x, r \oplus t_j) = 1$.

Note that there are only polynomial many j values to be checked, so these can be checked directly by the machine in polynomial time. $\square$