Notes on Complexity Theory Last updated: October, 2005

Handout 7

Jonathan Katz

1 More on Randomized Complexity Classes

Reminder: so far we have seen $\mathcal{RP}, \mathsf{co}\mathcal{RP}$, and $\mathcal{BPP}$. We introduce two more time-bounded randomized complexity classes: $\mathcal{ZPP}$ and $\mathcal{PP}$.

1.1 $\mathcal{ZPP}$

$\mathcal{ZPP}$ may be defined in various ways; we will pick one arbitrarily and state the others (and their equivalence) as claims.

Definition 1 Class $\mathcal{ZPP}$ consists of languages L for which there exists a PPT machine M which is allowed to output "1", "0", or "?" and such that, for all x:

$$\Pr[M(x) = \text{“?”}] \leq \frac{1}{2} \quad \text{and} \quad \Pr[M(x) = \chi_L(x) \vee M(x) = \text{“?”}] = 1.$$

That is, M always outputs either the *correct* answer or a special symbol "?" denoting "don't know", and it outputs "don't know" with probability at most half. $\diamondsuit$

Claim 1 *$\mathcal{ZPP} = \mathcal{RP} \cap \mathsf{co}\mathcal{RP}$ and hence $\mathcal{ZPP} \subseteq \mathcal{BPP}$.*

Claim 2 *$\mathcal{ZPP}$ consists of languages L for which there exists a machine M running in* **expected** *polynomial time and for which:*

$$\Pr[M(x) = \chi_L(x)] = 1.$$

(A machine runs in expected polynomial time if there exists a polynomial p such that for all x, the expected running time of $M(x)$ is at most $p(|x|)$.)

Summarizing what we have so far: $\mathcal{P} \subseteq \mathcal{ZPP} \subseteq \mathcal{RP} \subseteq \mathcal{BPP}$. However (somewhat surprisingly), it is currently believed that $\mathcal{P} = \mathcal{BPP}$.

1.2 $\mathcal{PP}$

$\mathcal{RP}, \mathsf{co}\mathcal{RP}, \mathcal{BPP}$, and $\mathcal{ZPP}$ represent possibly-useful relaxations of deterministic polynomial-time computation. The next class does not, as we will see.

Definition 2 $L \in \mathcal{PP}$ if there exists a PPT machine M such that:

$$\Pr[M(x) = \chi_L(x)] > \frac{1}{2}.$$

$\diamondsuit$

Note that requiring only $\Pr[M(x) = \chi_L(x)] \geq \frac{1}{2}$ makes the definition trivial, as it can be achieved by flipping a random coin. Note also that error reduction does *not* apply to $\mathcal{PP}$, since to increase the "gap" between acceptance and rejection to an inverse polynomial we might have to run our original algorithm an exponential number of times (and so we would no longer have a PPT algorithm).

Claim 3 $\mathcal{BPP} \subseteq \mathcal{PP} \subseteq \mathsf{PSPACE}$.

The first inclusion follows immediately from the definitions. The second inclusion follows since, given a PPT Turing machine M, we may enumerate all coins (and take a majority vote) in PSPACE.

The following shows that $\mathcal{PP}$ is too lax a definition:

Claim 4 $\mathcal{NP} \subseteq \mathcal{PP}$.

Proof Let $L \in \mathcal{NP}$, and assume that inputs $x \in L$ have witnesses of size exactly $p(|x|)$ (by padding, this is w.l.o.g.). Consider the following $\mathcal{PP}$ algorithm for L on input x: with probability $1/2 - 2^{-p(|x|)}/4$, accept. Otherwise, choose a string $w \in \{0,1\}^{p(|x|)}$ at random and accept iff w is a witness for x. Clearly, if $x \notin L$ then the probability that this algorithm accepts is less than $1/2$. On the other hand, if $x \in L$ then there is at least one witness and so the probability of acceptance is at least:

$$\frac{1}{2} - 2^{-p(|x|)}/4 + 2^{-p(|x|)}/2 > \frac{1}{2}.$$

■

2 $\mathcal{BPP}$ in Relation to Deterministic Complexity Classes

$\mathcal{BPP}$ is currently considered to be the "right" notion of what is "efficiently computable." It is therefore of interest to see exactly how powerful this class is.

2.1 $\mathcal{BPP} \subseteq \mathcal{P}/\text{poly}$

We claimed in an earlier lecture that $\mathcal{P}/\text{poly}$ provides an *upper* bound to efficient computation. We show that this is true with regard to $\mathcal{BPP}$.

Theorem 5 $\mathcal{BPP} \subseteq \mathcal{P}/\text{poly}$.

Proof Let $L \in \mathcal{BPP}$. We know that there exists a probabilistic polynomial-time Turing machine M and a polynomial p such that M uses a random tape of length $p(|x|)$ and $\Pr_r[M(x;r) \neq \chi_L(x)] < 2^{-2|x|^2}$. An equivalent way of looking at this is that for any n and each $x \in \{0,1\}^n$ there is a set of "bad" coins for x (for which $M(x)$ returns the wrong answer), but the size of this "bad" set is smaller than $2^{p(n)} \cdot 2^{-2n^2}$. Taking the union of these "bad" sets over all $x \in \{0,1\}^n$, we find that the total number of random coins which are "bad" for *some* $x \in \{0,1\}^n$ is at most $2^{p(n)} \cdot 2^{n-2n^2} < 2^{p(n)}$. In particular, for each n there exists at least one random tape $r_n^* \in \{0,1\}^{p(n)}$ which is "good" for *every* $x \in \{0,1\}^n$ (in fact, there are many such random tapes). If we let the sequence of "advice strings" be exactly these $\{r_n^*\}_{n \in \mathbb{N}}$, we obtain the result of the theorem. ■

2.2 $\mathcal{BPP}$ is in the Polynomial Hierarchy

We noted earlier that it is not known whether $\mathcal{BPP} \subseteq \mathcal{NP}$. However, we can place $\mathcal{BPP}$ in the polynomial hierarchy. Here, we give two different proofs that $\mathcal{BPP} \subseteq \Sigma_2 \cap \Pi_2$.

2.2.1 Lautemann's Proof

We first prove some easy propositions. For $S \subseteq \{0,1\}^m$, say S is *large* if $|S| \geq (1 - \frac{1}{m}) \cdot 2^m$. Say S is *small* if $|S| < \frac{2^m}{m}$. Finally, for a string $z \in \{0,1\}^m$ define $S \oplus z \stackrel{\text{def}}{=} \{s \oplus z \mid s \in S\}$. We first prove:

Proposition 6 *Let $S \subseteq \{0,1\}^m$ be small. Then for all $z_1, \ldots, z_m \in \{0,1\}^m$ we have $\bigcup_{i=1}^m (S \oplus z_i) \neq \{0,1\}^m$.*

This follows easily by counting. On the one hand, $|\{0,1\}^m| = 2^m$. On the other hand, for any $z_1, \ldots, z_m$ we have

$$\begin{aligned} \left| \bigcup_{i=1}^m (S \oplus z_i) \right| &\leq \sum_{i=1}^m |S \oplus z_i| \\ &= m \cdot |S| \ < \ 2^m. \end{aligned}$$

Furthermore:

Proposition 7 *If S is large, then:*

$$\Pr_{z_1,\ldots,z_m \in \{0,1\}^m} \left[\bigcup_{i=1}^m (S \oplus z_i) = \{0,1\}^m \right] > 1 - \left(\frac{2}{m}\right)^m.$$

To see this, consider first the probability that some fixed y is not in $\bigcup_i (S \oplus z_i)$. This is given by:

$$\begin{aligned} \Pr_{z_1,\ldots,z_m \in \{0,1\}^m} [y \notin \bigcup_i (S \oplus z_i)] &= \prod_{i=1}^m \Pr_{z_i \in \{0,1\}^m} [y \notin S \oplus z_i] \\ &\leq \left(\frac{1}{m}\right)^m. \end{aligned}$$

Applying a union bound over all $y \in \{0,1\}^m$, we see that the probability that there *exists* a $y \in \{0,1\}^m$ which is not in $\bigcup_i (S \oplus z_i)$ is at most $\frac{2^m}{m^m}$. The proposition follows.

Given $L \in \mathcal{BPP}$, there exist a polynomial m and an algorithm M such that M uses $m(|x|)$ random coins and errs with probability less than $1/m(|x|)$. For any $x \in \{0,1\}^n$, let $S_x \subseteq \{0,1\}^{m(|x|)}$ denote the set of random coins for which $M(x;r)$ outputs 1. Thus, if $x \in L$ (and letting $m = m(|x|)$) we have $|S_x| \geq (1 - \frac{1}{m}) \cdot 2^m$ while if $x \notin L$ then $|S_x| < \frac{2^m}{m}$. This leads to the following Σ_2 characterization of L:

$$x \in L \ \Leftrightarrow \ \exists z_1, \ldots, z_m \in \{0,1\}^m \ \forall y \in \{0,1\}^m : y \in \bigcup_{i=1}^m (S_x \oplus z_i).$$

Note that the desired condition can be efficiently verified by checking whether

$$\bigvee_i M(x; y \oplus z_i).$$

We conclude that $\mathcal{BPP} \subseteq \Sigma_2$.

Using the fact that $\mathcal{BPP}$ is closed under complement gives the claimed result.

2.2.2 The Sipser-Gács Proof

The proof in the previous section is a bit easier than the one we present here, but the technique here is quite useful. First, some notation. Let $h : \{0,1\}^m \to R$ be a function, let $S \subseteq \{0,1\}^m$, and let $s \in S$. We say that h *isolates* s if $h(s') \neq h(s)$ for all $s' \in S \setminus \{s\}$. We say a collection of functions $H \stackrel{\text{def}}{=} \{h_1, \ldots, h_\ell\}$ isolates s if there exists an $h_i \in H$ which isolates s. Finally, we say H isolates S if H isolates every element of S.

Proposition 8 *Let $\mathcal{H}$ be a family of pairwise-independent hash functions mapping $\{0,1\}^m$ to some set R. Let $S \subseteq \{0,1\}^m$. Then*

- *If $|S| > m|R|$ then:*

$$\Pr_{h_1,\ldots,h_m \in \mathcal{H}}[\{h_1, \ldots, h_m\} \text{ isolates } S] = 0.$$

- *If $|S|^{m+1} \leq |R|^m$ then:*

$$\Pr_{h_1,\ldots,h_m \in \mathcal{H}}[\{h_1, \ldots, h_m\} \text{ isolates } S] > 0.$$

Proof For the first part of the proposition, note that each hash function h_i can isolate at most $|R|$ elements. So m hash functions can isolate at most $m|R|$ elements. Since $|S| > m|R|$, the claim follows.

For the second part, assume S is non-empty (if S is empty then the theorem is trivially true). Consider the probability that a particular $s \in S$ is *not* isolated:

$$\begin{aligned}
\Pr_{h_1,\ldots,h_m \in \mathcal{H}}[\{h_1, \ldots, h_m\} \text{ does not isolate } s] &= \left(\Pr_{h_i \in \mathcal{H}}[h_i \text{ does not isolate } s]\right)^m \\
&\leq \left(\sum_{s' \in S \setminus \{s\}} \Pr_h[h(s') = h(s)]\right)^m \\
&< \left(\frac{|S|}{|R|}\right)^m.
\end{aligned}$$

Summing over all $s \in S$, we see that the probability that S is not isolated is less than $\frac{|S|^{m+1}}{|R|} \leq 1$. This gives the stated result. ■

Let $L \in \mathcal{BPP}$. Then there exists a machine M using $m(|x|)$ random coins which errs with probability less than $1/4m(|x|)$. Let $x \in \{0,1\}^n$, set $m = m(|x|)$, and define S_x as in the previous section. Set $|R| = 2^m/2m$. Note that if $x \in L$ then $|S_x| \geq 3 \cdot 2^m/4 > m \cdot |R|$. On

the other hand, if $x \notin L$ then $|S_x| < 2^m/4m$ and so $\frac{|S_x|^{m+1}}{|R|^m} < 1/4m$. The above proposition thus leads to the following Π_2 characterization of L:

$$x \in L \Leftrightarrow \forall h_1, \ldots, h_m \in \mathcal{H}\ \exists s, s_1, \ldots, s_m : \bigwedge_i ((s \neq s_i) \wedge (h_i(s) = h_i(s_i)) \wedge (s, s_i \in S_x))$$

(note that membership in S_x can be verified in polynomial time, as in the previous section). Closure of $\mathcal{BPP}$ under complement gives the stated result.

3 Randomized Space Classes

An important note regarding randomized space classes is that we do not allow the machine to store its previous random coin flips "for free" (it can, if it chooses, write its random choice(s) on its work tape). If we consider the model in which a randomized Turing machine is simply a Turing machine with access to a random tape, this implies that we allow only *unidirectional* access to the random tape.

There is an additional subtlety regarding the definition of randomized space classes: we need to also bound the running time. In particular, we will define RSPACE($s(n)$) as follows:

Definition 3 A language L is in RSPACE($s(n)$) if there exists a randomized Turing machine M using $s(n)$ space **and** $2^{O(s(n))}$ **time** and such that

$$x \in L \Rightarrow \Pr[M(x) = 1] \geq \frac{1}{2} \quad \text{and} \quad x \notin L \Rightarrow \Pr[M(x) = 0] = 1.$$

$\diamondsuit$

Without the time restriction, we get a class which is too powerful:

Proposition 9 *Define* RSPACE$'$ *as above, but without the time restriction. Then for any space-constructible* $s(n) \geq \log n$ *we have* RSPACE$'(s(n)) =$ NSPACE$(s(n))$.

Proof (Sketch) Showing that RSPACE$'(s(n)) \subseteq$ NSPACE$(s(n))$ is easy. We turn to the other direction. The basic idea is that, given a language $L \in$ NSPACE$(s(n))$, we construct a machine which on input x guesses valid witnesses for x (where a witness here is an accepting computation of the non-deterministic machine on input x). Since there may only be a single witness, we guess a doubly-exponential number of times. This is where the absence of a time bound makes a difference.

In more detail, given L as above we know that any $x \in L$ has a witness (i.e., an accepting computation) of length at most $\ell(n) = 2^{O(s(n))}$. Assuming such a witness exists, we can guess it with probability at least $2^{-\ell(n)}$ (and verify whether we have a witness or not using space $O(s(n))$). Equivalently, the expected number of times until we guess the witness is $2^{\ell(n)}$. The intuition is that if we guess $2^{\ell(n)}$ witnesses, we have a good chance of guessing a correct one. Looking at it in that way, the problem boils down to implementing a counter that can count up to $2^{\ell(n)}$. The naive idea of using a standard $\ell(n)$-bit counter will not work, since $\ell(n)$ is exponential in $s(n)$! Instead, we use a *randomized* counter: each time after guessing a witness, flip $\ell(n)$ coins and if they are all 0 stop; otherwise, continue. This can be done using a counter of size $\log \ell(n) = O(s(n))$.

Note that the Turing machine thus defined may run for infinite time; however, the probability that it does so is 0. In any case, it never uses more than $s(n)$ space, as required. Furthermore (using the fact that infinite runs occur with probability 0) the machine satisfies

$$x \in L \Rightarrow \Pr[M(x) = 1] \geq \frac{1}{2} \quad \text{and} \quad x \notin L \Rightarrow \Pr[M(x) = 0] = 1.$$

■

3.1 $\mathcal{RL}$

Define $\mathcal{RL} = \text{RSPACE}(\log n)$. We show that *undirected* graph connectivity is in $\mathcal{RL}$ (here, we are given an undirected graph and vertices s, t and asked to decide whether there is a path from s to t). This follows easily from the following result:

Theorem 10 *Let G be an n-vertex undirected graph, and s an arbitrary vertex in G. A random walk of length $4n^3$ beginning at s visits all vertices in the connected component of s with probability at least $1/2$.*

Proof In the next lecture we will discuss Markov chains and random walks on undirected graphs, and will show that if G is non-bipartite then for any edge (u, v) in the connected component containing s, the expected time to move from vertex u to vertex v is at most $2|E| + n \leq n^2$. (Note that if G is bipartite, we can make it non-bipartite by "mentally" adding n self-loops; the expected time to move from u to v is then at most $2|E| + n$ as claimed. But taking a random walk in the actual graph (without self-loops) can only result in a lower expected time to move from u to v.)

Consider any spanning tree of the connected component containing s; this will contain $n - 1$ edges. Considering any traversal of this spanning tree (which traverses fewer than $2n$ edges), we see that the expected time to reach every vertex in the connected component is at most $2n \cdot n^2 = 2n^3$. Taking a random walk for twice this many steps means we will reach the entire component with probability at least half. ■

We remark that the analogous result does *not* hold for *directed* graphs. (If it did, we would have $\mathcal{RL} = \mathcal{NL}$ which is considered unlikely. To be clear: it is possible that some other algorithm can solve directed connectivity in $\mathcal{RL}$, but the above algorithm does not.)

Next time, we will see another application of random walks to solving 2-SAT.

Bibliographic Notes

Sections 1 and 3 are adapted from [2, Lecture 7], and Section 2.2 from [1, Lectures 19–20].

References

[1] J.-Y. Cai. Scribe notes for CS 810: Introduction to Complexity Theory. 2003.

[2] O. Goldreich. Introduction to Complexity Theory (July 31, 1999).