ITCS:CCT09 : Computational Complexity Theory May 11, 2009

Lecture 17

Lecturer: Jayalal Sarma M.N. *Scribe: Hao Song*

In this lecture, we will continue our discussion on randomization.

1 BPP and the Polynomial Hierarchy

In this section, we will prove that the complexity class BPP is actually contained in PH, more specifically, we will prove that BPP is contained in $\Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$.

The following sketch illustrates some of the containment relationships about randomized complexity classes we've learnt so far

Theorem 1 $\mathsf{BPP} \subseteq \Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$

Proof From the definition of BPP, it's trivial to show that BPP = coBPP. Thus we only have to show that $\mathsf{BPP} \subseteq \Sigma_2^{\mathsf{P}}$ and get $\mathsf{BPP} = \mathsf{coBPP} \subseteq \Pi_2^{\mathsf{P}}$ as an immediate corollary.

From the definition of BPP, we know that if a language L is in BPP, then there's a PTM $\mathcal{M}$ which uses up to m random bits such that

$$\begin{aligned} x \in \mathsf{L} &\Rightarrow Pr_{y\in\{0,1\}^m}[\mathcal{M}(x,y)=1] \geq 1 - \frac{1}{2m} \\ x \notin \mathsf{L} &\Rightarrow Pr_{y\in\{0,1\}^m}[\mathcal{M}(x,y)=1] \leq \frac{1}{2m} \end{aligned}$$

Let's define

$$S_x \stackrel{\text{def}}{=} \{y \ : \ \mathcal{M}(x,y)=1\}$$

then the above definition translates to

$$\begin{aligned} x \in \mathsf{L} &\Rightarrow |S_x| \geq 2^m(1 - \frac{1}{2m}) \\ x \notin \mathsf{L} &\Rightarrow |S_x| \leq 2^m \frac{1}{2m} \end{aligned}$$

If we consider a bitstring $u \in \{0,1\}^m$, we can define the co-set of S_x induced by u to be $u \oplus S_x \stackrel{\text{def}}{=} \{u \oplus v \ : \ v \in S_x\}$. Then, intuitively, $|S_x|$ is large if and only if we can cover the whole space $\{0,1\}^m$ with only a "small" number of co-sets of S_x. To make that statement concrete, we claim that

$$x \in \mathsf{L} \Leftrightarrow \exists u_1, u_2, \dots u_m \in \{0,1\}^m \text{ s.t. } \bigcup_{i=1}^{m} (u_i \oplus S_x) = \{0,1\}^m \tag{1}$$

The "backward" direction of this claim can be trivially proved using a counting argument. Thus we will only prove the "forward" direction here. We will choose $u_1, \dots, u_m$ independently and uniformly at random from $\{0,1\}^m$, and say that

$$Pr_{u_1,\dots,u_m}[\exists r \in \{0,1\}^m \text{ s.t. } r \notin \bigcup_{i=1}^{m} (u_i \oplus S_x)] < 1 \tag{2}$$

To prove this, consider any $r \in \{0,1\}^m$, let's compute the probability that r is not covered by all those m co-sets of S_x.

$$\begin{aligned} Pr_{u_1,\dots,u_m}[r \text{ is not covered}] &= Pr_{u_1,\dots,u_m}[\forall i \ r \notin u_i \oplus S_x] \\ &= Pr_{u_1,\dots,u_m}[\forall i \ u_i \notin r \oplus S_x] \\ &= (\frac{1}{2m})^m \\ &< \frac{1}{2^m} \end{aligned}$$

Because there're 2^m different r's, by the *union bound*, we can say that the probability that any of them is not covered is (strictly) less than 1, hence equation (2). This means that there is at least one choice of $u_1, \dots, u_m$ such that the m co-sets of S_x induced by $u_1, \dots, u_m$ cover the whole space of $\{0,1\}^m$.

Further note that the fact that a bitstring $r \in \{0,1\}^m$ is covered by a co-set $u_i \oplus S_x$ can be equivalently stated as $r \oplus u_i \in S_x$, or $\mathcal{M}(x, r \oplus u_i) = 1$. Thus we can rephrase equation (1) as

$$\begin{aligned} x \in \mathsf{L} \quad \Leftrightarrow \quad & \exists u_1, \dots, u_m \in \{0,1\}^m \\ & \text{s.t. } \forall r \in \{0,1\}^m \bigvee_{i=1}^{m} \mathcal{M}(x, r \oplus u_i) = 1 \end{aligned}$$

This matches the alternating quatifiers formulation of Σ_2^{P} exactly, which means $\mathsf{BPP} \subseteq \Sigma_2^{\mathsf{P}}$. This concludes the prove. ■

This corollary follows immediately from the above theorem

Corollary 2 $\mathsf{NP} = \mathsf{P} \Rightarrow \mathsf{BPP} = \mathsf{P}$

We don't believe that NP is equal to P, but a lot of people do believe that $\mathsf{BPP} = \mathsf{P}$.

2 Problems about BPP

2.1 Is There a Complete Problem for BPP?

A natural question to ask about the BPP class is if there's a problem that is BPP-complete. Unfortunately, we don't know any thus far. One reason for this difficulty is that the defining property of BPTIME machines is *semantic*, namely, that for *every* string they either accept with probability at least 2/3 or reject with probability at least 1/3. Given the description of a PTM, even testing whether it has property is *undecidable*. By contrast, the defining property of an NTM is *syntactic*: given a string it is easy to determine if it is a valid encoding of an NTM. Completeness seems easier to define for syntactically defined classes than for semantically defined ones.

We do know a language L that is BPP-hard

$$\mathsf{L} \overset{\text{def}}{=} \{\langle M, x\rangle \ : \ Pr[M(x) = 1] \geq 2/3\}$$

But it is not known to be in BPP.

2.2 Does BPTIME Have a Hierarchy Theorem?

Unlike the DTIME and NTIME class families, we don't know about a hierarchy theorem for the BPTIME family, at least for now. The problem is that our proves for the hierarchy theorems we've known so far are all based on the diagonalization technique, which, in turn, relies on the existence of an enumeration of the corresponding class of machines. But because of the same reason we explained above, it is unlikely that such an enumeration exists.

3 Randomized Reductions

Definition 3 *We say that there is a* randomized reduction *from language* A *to language* B *if there is a PTM $\mathcal{M}$ such that*

$$Pr_y[x \in \mathsf{A} \ \Leftrightarrow \ \mathcal{M}(x, y) \in \mathsf{B}] \geq \frac{2}{3}$$

And if the PTM runs in polynomial time, we say that $\mathsf{A} \leq_r^{\mathsf{P}} \mathsf{B}$.

This definition of *randomized reduction* is modelled after the complexity class BPP. We can similarly define randomized reductions modelled after RP, coRP. Also note that this kind of randomized reduction is analogous to the *many-one* reductions introduced in previous lectures.

People often define complexity classes based on randomized reductions. For example, the complexity class $\mathsf{BP} \cdot \mathsf{NP}$ is defined as

$$\mathsf{BP} \cdot \mathsf{NP} \quad \overset{\text{def}}{=} \quad \{\mathsf{L} \ : \ \mathsf{L} \leq_r^{\mathsf{P}} \mathsf{3SAT}\}$$

we claim that

Proposition 4 $\mathsf{coNP} \subseteq \mathsf{BP} \cdot \mathsf{NP}$

4 Randomized Circuits

We can also introduce randomized inputs into other computation models, e.g. circuits. The randomzied analogue of the classical NC circuit class, RNC, can be defined as follows

Definition 5 *A language L is said to be in RNC, if there are three positive constants c, d, r such that L can be decided by a family of circuits $\{C_n\}$ where C_n has size $O(n^c)$ and depth $O(\log^d n)$, and takes, in addition to a length n instance x of L, a random string of length n^r as input. And the output of the circuit should be correct with probability at least $\frac{2}{3}$ (with respect to a uniform distribution of the random strings) for every instance of L.*

We know that

Proposition 6 $\mathsf{RNC} \subseteq$ *non-uniform* NC

As an example of a problem in the RNC class, let's consider the *Perfect Matching* problem, abbreviated as PM. This problem is to determine if there is a perfect matching in a given bipartite n by n graph G. It's easy to see that this graph can be represented by a n by n adjacency matrix $M = (a_{i,j})$ $(i, j \in [n])$.

$$a_{i,j} = \begin{cases} 1 & \text{if the } i\text{th vertex on one side is adjacent to the } j\text{th vertex on the other side} \\ 0 & \text{otherwise} \end{cases}$$

We can construct a new matrix $M' = (a'_{i,j})$ on a set of n^2 variables $x_{i,j}$ $(i, j \in [n])$ as $a'_{i,j} = a_{i,j} x_{i,j}$. We claim that

Claim 7 *G has a perfect matching if and only if the determinant of M' is a non-zero polynomial.*

Note that

Lemma 8 *The determinant of a matrix can be computed in* NC^2.

Also note that the problem of testing if a given polynomial is identically zero can be solved efficiently with random sampling, based on the following result

Lemma 9 *Given a field $\mathcal{F}$, a non-zero polynomial p of n variables and degree d, and a finite subset of the field $\mathcal{S} \subseteq \mathcal{F}$, we have*

$$Pr_{a_1,\ldots,a_n \in \mathcal{S}}[p(a_1, a_2, \ldots, a_n) = 0] \leq \frac{d}{|\mathcal{S}|}$$

Thus we conclude that PM can be solved in RNC.

Regarding the relationships between RNC and the previously discussed complexity classes, we don't yet know if RNC is in P, but we do know that RNC is in BPP.

5 Space Bounded Randomization

In the previous sections, we mainly talk about time-bounded randomization. In this section, we will start to talk about space bounded randomization. For a function $s(\cdot)$, we say that a PTM uses space $s(\cdot)$ if in the computational tree of this machine on an input string of length n

- in every computational path, the machine touches no more than $s(n)$ cells on the working tape.
- every computational path is of length $O(2^{s(n)})$.

Now we can define the complexity classes RL and BPL as the *logspace* counterparts of the classes BPP and RP we defined in previous lectures.

The following sketch summarizes the state of our knowledge about the relationships between related complexity classes

We will prove in the following lectures that BPL is actually contained in P.