6.841/18.405J Advanced Complexity Theory March 3, 2003

Lecture 7: BPP $\subseteq$ PH, Circuit Lower Bounds

Instructor: Prof. Madhu Sudan *Scribe: Jonathan Derryberry*

1 BPP $\subseteq$ PH

A review of the currently known relationships among relevant complexity classes appears in Figure 1.

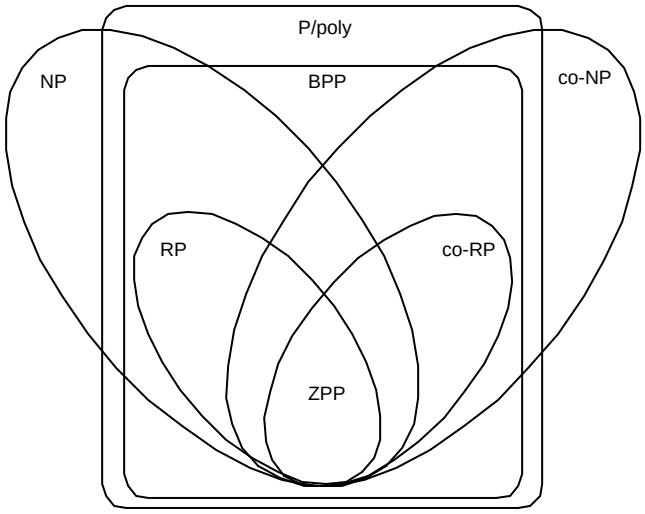

Figure 1: The currently known relationships among complexity classes.

The purpose of this part of the lecture is to demonstrate that BPP $\subseteq$ PH, in particular that BPP $\subseteq \Sigma_2^{\mathrm{P}}$. Before proceeding with the proof, we will get an idea of what we are trying to do. For an arbitrary language $L \in$ BPP, we are trying to show that a string w is in L exactly when $\exists x_1, \ldots, x_n \forall y_1, \ldots, y_n P(x_1, \ldots, x_n, y_1, \ldots, y_n) = 1$ for some polynomial time computation P. In other words, the "x player" is capable of finding a sequence of xs such that for any sequence of ys that the "y player" chooses, the polynomial time "audience" will accept.

Suppose M is a BPP machine for L. The above characterization of membership in Σ_2^{P} raises three questions that the proof of BPP $\subseteq \Sigma_2^{\mathrm{P}}$ must answer in constructing such a "debate system" using M:

1. What should the x player do?
2. What should the y player do?
3. What should P be?

Using M, on input w (with $|w| = n$) consider all possible random strings z (with $|z| = m = poly(n)$) that could be used. Define a "bad string" to be a string z that causes M to make the wrong decision about whether $w \in L$ (e.g. $M(w, z) = 0$ but $w \in L$). Note that because M is a BPP machine for L, it

is the case that if $w \in L$ then $\Pr[M(w,z) = 1] > 1 - 2^{-n}$, and if $w \notin L$ then $\Pr[M(w,z) = 1] < 2^{-n}$. Thus, there are very few bad random strings z on a particular input w.

Let us consider partitioning the universe of all possible random strings of length m into two halves. One half contains only good strings and one half contains all of the bad strings (See Figure 2). If

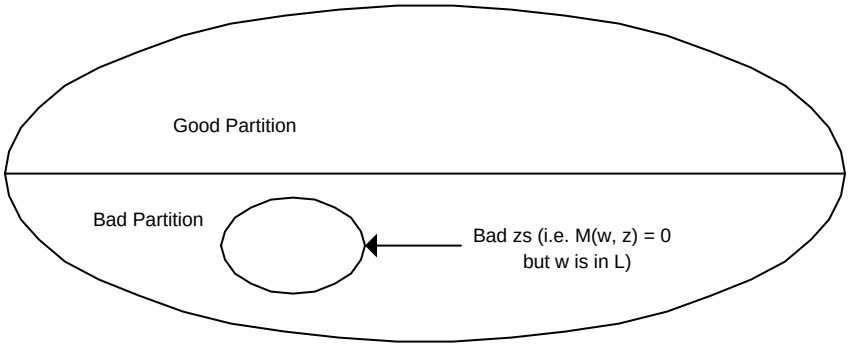

Figure 2: A schematic of a partition of the universe of random strings of length m.

the random string z falls in the good partition then M gives the correct answer, while if z falls in the bad partition there is a small probability of error. Thus, if an efficiently computable bijection $\pi : \{0,1\}^m \mapsto \{0,1\}^m$ is created between the two halves, then the results of both $M(w,z)$ and $M(w,\pi(z))$ can be examined with assurance that one of them is correct and one may be incorrect with low probability. Thus, if $w \in L$ then $M(w,z) = 1$ or $M(w,\pi(z)) = 1$.

Note that the x player can announce this bijection π in its round, and the y player can check whether π is, in fact, a bijection by testing all possible inputs to the bijection and exhibiting a pair that map to the same value to the audience if x tried to cheat by announcing a function that was not a bijection.

One problem with this is that it is not clear that such an efficiently computable bijection exists. One candidate bijection is the family of bijections

$$\pi_r : z \mapsto z \oplus r,$$

where $r \in \{0,1\}^m$. It is clear that π_r is a bijection and is efficiently computable; however it is not clear that it creates a good partition. Nevertheless, on average we expect that using π_r for a particular random r will cause $\Pr[M(w,z) \vee M(w,\pi_r(z)) = 0]$ to be much smaller than $\Pr[M(w,z) = 0]$ given that $w \in L$ because the probability that both z and $\pi_r(z)$ are bad is less than $2^{-n} \cdot 2^{-n}$, where n is the length of w.

Clearly, this represents an improvement so let us strengthen the improvement by choosing $r_1, \ldots, r_l$ at random and fixing z to some random value. Continuing the argument from above, we have

$$\Pr[M(w,\pi_{r_1}(z)) \vee \cdots \vee M(w,\pi_{r_l}(z)) = 0] < (2^{-n})^l,$$

which is less than 2^{-m} for some $l \leq m$. Note that this probability is so small that it is less than the probability of choosing a particular m bit string at random. Also, note that if $w \notin L$ then the probability of making an error has increased, but only linearly in l because

$$\Pr[M(w,\pi_{r_1}(z)) \vee \cdots \vee M(w,\pi_{r_l}(z)) = 1] < l \cdot 2^{-n},$$

Now, to answer the three questions from above, consider what happens if

1. The x player uses the strings $r_1, \ldots, r_l$.

2. The y player uses all values of z.

3. The audience uses $M(w, \pi_{r_1}(z)) \vee \cdots \vee M(w, \pi_{r_l}(z))$.

That is, we consider the formula

$$\exists r_1, \ldots, r_l \forall z M(w, \pi_{r_1}(z)) \vee \cdots \vee M(w, \pi_{r_1}(z)). \tag{1}$$

To see that Equation 1 proves that $L \in \Sigma_2^{\mathrm{P}}$, note that $M(w, \pi_{r_i}(z))$ is a polynomial time computation and it is repeated l times, which is also polynomial in n. Also, note that it is correct because:

- If $w \in L$, then the probability that for random $r_1, \ldots, r_l$ $M(w, \pi_{r_1}(z)) \vee \cdots \vee M(w, \pi_{r_l}(z))$ gives the wrong answer is so small that less than one potential z (member of $\{0,1\}^m$) causes an error on average. Thus, there must be some choice of $r_1, \ldots, r_l$ such that no error occurs for any z. That is,

$$w \in L \implies \exists r_1, \ldots, r_l \forall z M(w, \pi_{r_1}(z)) \vee \cdots \vee M(w, \pi_{r_1}(z)) = 1.$$

- If $w \notin L$, then for random $r_1, \ldots, r_l$ $\Pr[M(w, \pi_{r_1}(z)) \vee \cdots \vee M(w, \pi_{r_1}(z)) = 1] < l \cdot 2^{-n}$ as shown above. Thus, for *any* setting of $r_1, \ldots, r_l$ there is some z (in fact, there are many) such that $M(w, \pi_{r_1}(z)) \vee \cdots \vee M(w, \pi_{r_1}(z)) = 0$. In other words,

$$w \notin L \implies \forall r_1, \ldots, r_l \exists z M(w, \pi_{r_1}(z)) \vee \cdots \vee M(w, \pi_{r_1}(z)) = 0.$$

  This is just another way of stating

$$w \notin L \implies \neg\exists r_1, \ldots, r_l \forall z M(w, \pi_{r_1}(z)) \vee \cdots \vee M(w, \pi_{r_1}(z)) = 1.$$

2 Circuit Lower Bounds

To motivate the discussion of circuit lower bounds, consider the question of whether it is possible to get rid of the randomness in randomized algorithms and still have them recognize the same languages. Randomness can be simulated by constructing or computing strings that look random to polynomial time machines or, say, time n^2 machines. An important component of this progress involves proving *circuit lower bounds.*

Recall that a circuit is a directed acyclic graph with:

- n *input* nodes, which have in-degree 0, arbitrary out-degree, and are labeled $x_1, \ldots, x_n$
- 1 *output* node, which has arbitrary in-degree and out-degree 0
- many *internal* nodes, which have 3 types of functions:
  1. NOT, which has in-degree 1 and arbitrary out-degree
  2. OR, which has arbitrary in-degree and arbitrary out-degree
  3. AND, which has arbitrary in-degree and arbitrary out-degree

For our purposes, there are two important measures of circuits:

1. The SIZE of a circuit is the number of edges in its graph.
2. The DEPTH of a circuit is the length of the longest path in its graph.

One universal bound on circuit size is that for all functions $f : \{0,1\}^n \mapsto \{0,1\}$, there is a circuit of size $n2^n$ that computes f. To see that this is true, consider the circuit that has the entire table of values for f encoded as a sum of products (DNF).

Proposition 2-1. *There exists a function f that cannot be computed by a circuit of size $2^n/n$.*

The typical function for which lower bounds are proven is $f(h,x) = h(x)$, which splits the input into a function h and an input x for h. In this way, diagonalization can be used to prove that there is some $\{f_n\}$, computable in PSPACE that requires a size $n \log n$ circuit.

If C is a circuit of size s and depth d that computes $x_1 \oplus \cdots \oplus x_n$ (i.e. parity) then $s \geq 2^{n^{\Omega(1/d)}}$.