Lecture 12, February 19, 2015

We began this lecture with an alternative formulation of P/poly, in terms of *Turing Machines that take advice*. See Definition 6.16 and Theorem 6.18.

Although good lower bounds on the circuit complexity of specific boolean functions have proven very elusive, the following basic result of Shannon from 1949 establishes that some boolean functions require large circuits.

Theorem 1 *For every $n > 1$, there exists a function $f : \{0,1\}^n \longrightarrow \{0,1\}$ that cannot be computed by a circuit of size $\frac{2^n}{10n}$.*

Proof. Recall that a circuit is a directed acyclic graph with in-degree 2 and that its size is the number of nodes in it. So every circuit of size S can be represented by a bitstring of length at most $9S \log S$. For each of node i, $1 \leq i \leq S$, we must specify at most two predecessors, say v_{i1} and v_{i2}, and also the type of i, *i.e.*, one of input node, OR gate, AND gate, and NOT gate. Each of v_{i1} and v_{i2} can be represented by $\log S$ bits, and the type requires only two bits. (It seems as though $(2 \log S + 2)S$ bits would suffice for a circuit of size S, but the upper bound of $9S \log S$ given in the book is certainly valid.)

Thus, the total number of circuits of size $S = \frac{2^n}{10n}$ is at most

$$2^{9S \log S} = 2^{9 \cdot \frac{2^n}{10n} \cdot \log \frac{2^n}{10n}} = 2^{2^n (\frac{9}{10n} \cdot \log \frac{2^n}{10n})}.$$

Note that

$$\frac{9}{10n} \log \frac{2^n}{10n} < \frac{9}{10n} \log 2^n = \frac{9}{10n} \cdot n = \frac{9}{10} < 1.$$

Therefore, the number of circuits of size $\frac{2^n}{10n}$ is strictly less than the number of boolean functions on $\{0,1\}^n$, which is 2^{2^n}. ∎

Circuits of size $\frac{2^n}{10n}$ are considered "large" because of the following basic fact.

Theorem 2 *Any Boolean function on $\{0,1\}^n$ can be computed by a circuit of size $n(2^n + 1)$.*

Proof: Let f be a boolean function on $\{0,1\}^n$. Assume without loss of generality that f assumes the value 1 on at most half of its input vectors. If this is not the case, then we will build a circuit for $\neg f$ and just add one NOT gate in order to compute f. We first put into our circuit C for f exactly n input nodes $x_1, \ldots, x_n$; taking advantage of the fact that nodes in circuits may have unbounded fan-out (but fan-in at most two), we build at most 2^{n-1} subcircuits on these input nodes, one for each input vector on which f outputs 1.

The subcircuit C^j for $\epsilon^j = (\epsilon_1^j, \ldots, \epsilon_n^j)$ outputs 1 if and only if the input vector is equal to ϵ^j. So it contains a NOT gate v_i^j if and only if $\epsilon_i^j = 0$. The output of C^j is the n-way AND of either x_i (if $\epsilon_i^j = 1$) or v_i^j (if $\epsilon_i^j = 0$); this n-way AND can be computed with $n - 1$ AND gates of fan-in two. Thus, C^j contributes at most $2n - 1$ to the size of C.

To complete the construction of C, we must take the 2^{n-1}-way OR of the outputs of the C^j's and then perhaps add one more NOT gate as explained above. The 2^{n-1}-way OR can be computed using $2^{n-1} - 1$ OR gates of fan-in two. The size of C is thus at most

$$n + 2^{n-1} \cdot (2n - 1) + 2^{n-1} - 1 + 1 = n(2^n + 1). \qquad \blacksquare$$

We then turned to Chapter 7, on randomized computation. We began by reviewing Definitions 7.1, 7.2, and 7.6, as well as Theorem 7.10 – all were first presented in Lecture 1. We ended the class with a fundamental fact about the relationship between probabilistic, polynomial-time Turing Machines and polynomial-sized circuits.

Adleman's Theorem: BPP $\subseteq$ P/poly

Proof. Let L be a set in BPP. Recall that the Chernoff bounds on the tails of the binomial distribution ensure that there is a probabilistic polynomial-time machine M such that $M(x) = L(x)$ with probability at least $1 - 2^{-(n+1)}$. Let m be the maximum number of random bits that M uses on inputs of length n. So $m = \text{poly}(n)$, and M's output on input x is a function of x and a random string $r \in \{0,1\}^m$; this function of x and r is computable in deterministic polynomial time.

Fix a length n, and consider all inputs $x \in \{0,1\}^n$. We say that *r is bad for x* if M outputs the wrong answer on input x and random string r; otherwise, *r is good for x*. Because M's error probability is at most $2^{-(n+1)}$, the number of r's that are bad for any given x is at most $(2^m)/(2^{n+1})$. The total number of r's that are bad for *at least one* x is thus at most $(2^n) \cdot ((2^m)/(2^{(n+1)})) = 2^{m-1}$. (This maximum would be achieved if the set of r's that are bad for x_1 were disjoint from the set of r's that are bad for x_2, for all $x_1 \neq x_2$.) This means that there are $2^m - 2^{m-1} > 0$ strings r that are good for all $x \in \{0,1\}^n$.

Let r_n be a random string that is good for all $x \in \{0,1\}^n$. The circuit C_n that accepts elements of $L \cap \{0,1\}^n$ is "M on inputs of length n, with r_n hardcoded in," i.e., one that computes precisely the function that M computes on inputs of length n when it uses the random string r_n. The proof of Theorem 6.6 (P $\subseteq$ P/poly) shows that $\{C_n\}_{n \geq 1}$ is a polynomial-sized circuit family. ∎

Note that BPP is actually *properly* contained in P/poly. As we saw in Chapter 6, P/poly contains undecidable sets; on the other hand, everything in BPP is clearly decidable (in fact, clearly in PSPACE).