80240233: Complexity of Computation **Lecture 8**
ITCS, Tsinghua Univesity, Fall 2007 2 November 2007

Instructor: Andrej Bogdanov **Notes by:** Decheng Dai and Wei Yu

1 Parity is inapproximable by polynomials

The *depth* of a circuit is the length of the longest directed path from an input node to the output node.

Definition 1. AC^0 *is the class of languages computable by circuit families of constant depth, polynomial size, and whose gates have unbounded fanin and fanout.*

In the last lecture we started proving the following theorem:

Theorem 2. *Let* PARITY *be the parity function. That is, for every* $x \in \{0,1\}^n \cdot \mathrm{PARITY}(x_1, ..., x_n) = \sum_{i=1}^n x_i \pmod 2$. *Then* $\mathrm{PARITY} \notin \mathrm{AC}^0$.

In the last lecture we proved that for every $f : \{0,1\}^n \to \{0,1\}$, that can be computed by a circuit of size s and depth d can be approximated as

$$\Pr_x[p(x) = f(x)] \geq 0.99$$

where p is a polynomial of degree $O((\log s)^{2d})$. And we will prove that

Theorem 3. *For every f of degree $\sqrt{n}$,*

$$Pr[p(x) = \mathrm{PARITY}(x)] < 0.99$$

This will establish 2.

As the definition of $\mathrm{PARITY} : \{0,1\}^n \to \{0,1\}, \mathrm{PARITY}(x_1, ..., x_n) = \sum_{i=1}^n x_i \pmod 2$, we can define the function $\widetilde{\mathrm{PARITY}}$ similar to PARITY, which is the same function except that it is a mapping from $\{-1,1\}^n \to \{-1,1\}$.

Definition 4. $\widetilde{\mathrm{PARITY}} : \{-1,1\}^n \to \{-1,1\}$, $\widetilde{\mathrm{PARITY}}(x_1, ..., x_n) = \prod_{i=1}^n x_i$.

We can translate each function $p : \{0,1\}^n \to \{0,1\}$ to a function $\tilde{p} : \{-1,1\}^n \to \{-1,1\}$ via the formula

$$\tilde{p}(x_1, ..., x_n) = 1 - 2p(\frac{1-x_1}{2}, ..., \frac{1-x_n}{2}).$$

This has the effiect of replacing 0 with 1 and replacing 1 with -1.

Claim 5. *Every function $f : \{-1,1\}^n \to \mathbb{R}$ can be represented as*

$$f(x_1, ..., x_n) = \sum_{S \subseteq \{1,...,n\}} c_s (\prod_{i \in S} x_i)$$

where c_s is a real number.

Proof. By induction on n:

$$f(x_1, ..., x_n) = \frac{1+x_1}{2} \cdot f(1, x_2, ..., x_n) + \frac{1-x_1}{2} \cdot f(0, x_2, ..., x_n)$$

Because $f(i, x_2, ..., x_n)$ is a function from $\{0,1\}^{n-1}$ to $\{0,1\}$, we can apply the inductive hypothesis. The base case $n = 0$ is trivial. $\square$

Theorem 6. *For every polynomial p of degree $\sqrt{n}$,*

$$\Pr_{x \sim \{0,1\}^n}[p(x) = \widetilde{\text{PARITY}}(x)] < 0.99.$$

Proof. Suppose not, and fix a polynomial p such that

$$\Pr_{x \in \{0,1\}^n}[p(x) = \widetilde{\text{PARITY}}(x)] \geq 0.99.$$

Let $A = \{x : p(x) = \widetilde{\text{PARITY}}(x)\} \subseteq \{-1,1\}^n$. We will consider the set of all functions $f : A \to \mathbb{R}$ and "count" the number of such functions in two different ways. Counting functions from A to $\mathbb{R}$ does not quite make sense because there are infinitely many such functions, but the set of all such functions forms a linear space F over $\mathbb{R}$ of dimension $|A|$. So instead of counting functions we will be counting dimensions. On the one hand, we have

$$\dim F = |A| \geq 0.99 \cdot 2^n. \tag{1}$$

But since each function $f \in F$ can be writen as a polynomial, we can also write,

$$\begin{aligned}
f(x) &= \sum_{S \subseteq \{1,...,n\}} c_s \prod_{i \in S} x_i \\
&= \sum_{S:|S| \leq \frac{n}{2}} c_s \prod_{i \in S} x_i + \sum_{S:|S| > \frac{n}{2}} c_s \prod_{i \in S} x_i \\
&= \sum_{S:|S| \leq \frac{n}{2}} c_s \prod_{i \in S} x_i + \sum_{S:|S| > \frac{n}{2}} c_s \prod_{i \in S} x_i \cdot \widetilde{\text{PARITY}}(x) \cdot p(x) \\
&= \sum_{S:|S| \leq \frac{n}{2}} c_s \prod_{i \in S} x_i + \sum_{S:|S| > \frac{n}{2}} c_s \prod_{i \notin S} x_i \cdot p(x)
\end{aligned}$$

We can see that $\deg(\prod_{i \notin S} x_i) \leq \frac{n}{2}$ and $\deg(p(x)) \leq \sqrt{n}$, consequently $\deg(f) \leq \frac{n}{2} + \sqrt{n}$. So each function in F can be written as a polynomial of degree $n/2 + \sqrt{n}$ over the set A. Now the polynomials of degree at most $n/2 + \sqrt{n}$ also form a linear space P. One basis of this linear space consists of all monomials of degree at most $n/2 + \sqrt{n}$, and there are $\sum_{i=0}^{n/2+\sqrt{n}} \binom{n}{i}$ such monomials. So we have

$$\dim F \leq \dim P = \sum_{i=0}^{n/2+\sqrt{n}} \binom{n}{i} < 0.99 \cdot 2^n$$

for sufficiently large n (for instance using the central limit theorem). This contradicts (1). $\square$

2 Randomness versus determinism

We want to know that if NP $\not\subseteq$ P/poly, but all we can prove are things like PARITY $\not\subseteq$ AC^0. In the next few lectures we will see that proving circuit lower bounds is interesting not only because it seems a good way to make progress on proving P $\neq$ NP, but also – for much less obvious reasons – it is deeply connected to the question of how much randomness helps in making computations more efficient.

The best deterministic simulation we know of randomized algorithms for decision problems comes from the trivial fact BPP $\subseteq$ EXP. Is a subexponential, or even polynomial-time simulation, of BPP-type algorithm possible? Let's review some evidence for this problem. There were some instances in the past where people came up with an efficient randomized algorithm for some problem, only to discover later that the same problem can be solved deterministically. One example is the problem problem of determining if a number is prime. This is a coNP question. In the 1970s it was first shown that this problem is in coRP, then in RP as well. Many years later – in 2002 – primality testing was discovered to be in P.

This parable seems to give evidence that if we work hard enough, we can remove randomness from algorithms that appear to require it. Another example of this kind is testing for the existence of a perfect matching in a bipartite graph. This problem can be solved by a simple randomized algorithm. There is also a deterministic algorithm, though considerably more elaborate. (In this case, the deterministic algorithm was discovered first.)

To give contrasting evidence, here is an example of a problem that can be solved by a simple randomized polynomial-time algorithm, but for which the best known deterministic algorithm takes exponential time.

Polynomial Identity Testing (PIT) INPUT: An arithmetic formula $F(x_1, ..., x_n)$ with integer coefficients PROBLEM: Is F identically zero?

An *arithmetic formula* over the integers is an expression built up from the variables $x_1, \ldots, x_n$, the integers, and the arithmetic operations $'+'$ and $'\times'$, for instance:

$$(x_1 + 4 \times x_3 \times x_4) \times \big((x_2 - x_3) \times (x_2 + x_3)\big) - 3 \times x_2.$$

We say that F is identically zero, in short $F \equiv 0$, if when we expand the formula and carry out all cancellations we obtain 0.

A deterministic algorithm that comes to mind for this problem is to work out the symbolic expansion of the polynomial and check if everything cancels out. This might take exponential time. A clever alternative is to carefully choose a set of points $(a_1, \ldots, a_n) \in \mathbb{Z}^n$ and evaluate $F(a_1, \ldots, a_n)$. If F does not evaluate to zero, then we can safely answer "no". But if F always evaluates to zero, it might either be that $F \equiv 0$, or maybe $F \not\equiv 0$ but our choice of evaluation points was unlucky and we always ended up with the zero polynomial. It is not known how to choose evaluation points deterministically in a manner that rules out the second possibility. However, it won't be hard to show that a *random* choice of points is likely to work.

Theorem 7. PIT $\in$ coRP

Proof. Let m be the number of multiplications in F. Consider the following randomized algorithm for PIT:

```
A:  On input $F$,
    Choose values $a_1, \ldots, a_n$ independently at random from the set $\{1, \ldots, 2m\}$.
    Evaluate $b = F(a_1, \ldots, a_n)$ (over the integers).
    If $b \neq 0$, reject; otherwise, accept.
```

So, if $F \equiv 0$, this algorithm always accepts it. Now we show that if $F \not\equiv 0$, then the algorithm rejects F with probability 1/2. We will need the following theorem.

Theorem 8 (Schwarz-Zippel)**.** *For any nonzero polynomial p of degree d over the integers and every set $S \subseteq \mathbb{Z}$,*

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

In our case, F is a polynomial of degree at most m and S is a set of size $2m$, so the algorithm will detect that $F \not\equiv 0$ with probability at least 1/2. □

Proof of Theorem 8. We prove it by induction on n. If $n = 1$, $Pr_{a \in S}[p(a) = 0] \leq \frac{d}{|S|}$, that is, there are at most d values $a_1, ...a_d$, which satisfy $p(a_i) = 0$.

Suppose not, then exist distinct $a_1, ...a_{d+1}$, which satisfies $p(a_1) = ... = p(a_{d+1}) = 0$, so $\prod_{i=1}^{d}(x - a_i)$ divides p and $\prod_{i=1}^{d+1}(x - a_i)$ also divides p, which conflicts.

If $n > 1$, we have

$$P(x_1, ..., x_n) = x_1^{d_1} \cdot p_{d_1}(x_2, ..., x_n) + x_1^{d_1 - 1} \cdot p_{d_1 - 1}(x_2, ..., x_n) + ... + p_0(x_2, ..., x_n)$$

So,

$$\Pr[p(a_1, ..., a_n) = 0] \leq \Pr[P_{d_1}(a_2, ..., a_n) = 0] + \Pr[p(a_1, \ldots, a_n) = 0 \mid p_{d_1}(a_2, ..., a_n) \neq 0]$$

By induction hypothesis and case $n = 1$,

$$\Pr[p_{d_1}(a_2, ..., a_n) = 0] \leq (d - d_1)/|S|$$
$$\Pr[p(a_1, \ldots, a_n) = 0 \mid p_{d_1}(a_2, ..., an) \neq 0] \leq d_1/|S|.$$

So,

$$\Pr[p(a_1, ..., a_n) = 0] \leq \frac{d}{|S|}. \qquad \square$$

Since we do not know of any subexponential time deterministic algorithm for polynomial identity testing, it may seem reasonable to conjecture that this problems *requires* exponential time. But then a result of Impagliazzo and Wigderson gives us the unlikely consequence that BPP = EXP — namely *all* problems in exponential time, including SAT, MIN-FORMULA, etc. can be solved efficiently using randomness! This seems very unlikely. Thus even though we don't *know* of any subexponential algorithm for polynomial identity testing, theory tells us that such an algorithm is quite likely to exist. (In fact, theory gives us a very good candidate for such an algorithm, but one whose correctness relies on reasonable yet unproven complexity assumptions.)

This is an instance of a general result of Impagliazzo and Wigderson, which says that either BPP = EXP, or every BPP algorithm can be simulated in subexponential time. We won't prove this result. One of the principal ingredients in the proof — a statement that is quite interesting by itself — is the following theorem (that combines work by Nisan and Widgerson and Impagliazzo and Wigderson):

Theorem 9. *Suppose there is a decision problem L such that*

- *L can be decided by some deterministic Turing Machine in time $2^{c \cdot n}$ for some $c > 0$;*
- *L cannot be decided by any circuit family of size $2^{\delta \cdot n}$ for some $\delta > 0$,*

then BPP = P.

Roughly, if there exist a certain kind of problem (computable in time $2^{O(n)}$) that is hard for "small" circuits (of size $2^{\delta n}$), then randomness does not help. The assumption appears quite believable; so one way of eliminating randomness from algorithms goes by way of proving circuit lower bounds.