Yao's XOR Lemma: Additional Notes

Satyadev Nandakumar

August 18, 2026

Contents

1 Motivation

A *language* is a set of strings. A *complexity class* is a set of languages.

Randomized algorithms or probabilistic polynomials are quite easy to design when compared to deterministic polynomial-time algorithms. For problems like *Polynomial Identity Testing*, there are simple randomized bounded-error, polynomial-time algorithms, but we do not know any (deterministic) polynomial-time algorithms. Hence an important question in computer science is the following: can we remove the necessity of randomization from these algorithms?

The class of languages that have a bounded-error, probabilistic polynomial-time algorithm is called BPP. The question of whether all such algorithms have a deterministic polynomial-time solution, is the famous $P = BPP?$ question.

One standard approach to the P $=$ BPP? question is *derandomization* - *i.e.* techniques to remove randomness from a particular algorithm, or a whole class of algorithms in order to yield fast deterministic algorithms. For example, one could use a short "seed" consisting of truly random bits, and use it to generate

more pseudorandom bits which can then be used by the algorithm. The initial part of the course deals with the derandomization of BPP.

In our course, since we have to derandomize a whole *class* of algorithms, we need a fairly general approach to derandomization. Our general approach is roughly as follows. Keep in mind that the algorithms give yes/no answers, and do not compute arbitrary function values.

1.1 Outline of the approach

1. Pick a problem which is sufficiently hard in the worst-case.
2. Amplify the hardness so that it becomes hard on average. We will cover two separate approaches - the older one using Yao's XOR lemma, and the newer one using error-correcting codes.
3. Once we get a problem that is strongly hard on average, we use a pseudorandom generator to stretch approximately $\log n$ bits of a random seed to a longer "pseudorandom string" with length polynomial in n.
4. We can then brute-force try all random seeds in time $\text{poly}(n)$, simulate the randomized algorithm with all the pseudorandom outputs, and output the majority value.

The overall approach came quite close to fully derandomizing BPP. In step 1, we assume that there is a problem solvable in EXP - *i.e.* $O(2^{\text{poly}(n)})$ time - which cannot be solved by circuits having size polynomial in n. Based on this, finally we get full derandomization for BPP. We already know EXP has problems unsolvable in P. Instead of circuits, if step 1 had used algorithms, we would have the full result.

2 Circuits and hardness: definitions

A Boolean circuit, abstractly, is a directed acyclic graph having nodes of five types : input, output, OR, AND and NOT. The OR and AND gate have fan-in two, and the others have fan-in one. All gates have exactly one output.

The *size* of a Boolean circuit is the number of gates in the circuit. (There are more elaborate notions of size, but we do not need those in our course.) Instead of the running time of an algorithm, the hardness of a problem can be measured by the size of the circuit required to solve the problem.

Formally, in complexity theory, we consider a *circuit family* $\mathcal{C} = \langle C_1, C_2, \dots \rangle$, where for each n, the circuit C_n solves the problem at hand for all inputs of length n. When we consider problems of a different length m, we will use the circuit C_m from this family. Throughout the course, unless specifically mentioned, a circuit will mean a circuit family, and not an individual circuit.

A circuit (family) is said to have polynomial size if there is a polynomial q such that for every n, size of the circuit $C(n)$ is upper bounded by $O(q(n))$.

We now define what it means for a Boolean function to be *difficult* on average and in the worst case, for circuits (note: not difficult for algorithms!).

Definition 2.1 (Average-case hardness)**.** Let $0 < \rho < 1$. The *ρ-average-case circuit hardness* of a Boolean function $f_n : \{0,1\}^n \to \{0,1\}$, denoted $H^{\rho}_{\text{avg}}(f)$ is defined by

$$H^{\rho}_{\text{avg}}(f) = \max\{S \mid \text{ for every circuit } C_n \text{ of size S}, Pr_{x\sim U_n}[C_n(x) = f_n(x)] < \rho\}. \tag{1}$$

Thus, any circuit smaller than $H^{\rho}_{\text{avg}}(f)$ fails to compute f on more than $\rho 2^n$ number of strings.

Definition 2.2 (Worst-case hardness)**.** The *worst-case circuit hardness* of a Boolean function $f : \{0,1\}^n \to \{0,1\}$, denoted $H_{\text{w}}(f)$, $f_n : \{0,1\}^n \to \{0,1\}$ is $H^1_{\text{avg}}(f)$.

In the definition of worst-case hardness, we have to get $f(x)$ correct on *every* input x.

The larger the ρ, we must get a greater fraction of inputs correct. Hence we need larger circuits for larger ρ. Thus, $H^{\rho}_{\text{avg}}(f) \leq H_{\text{w}}(f)$.

3 Yao's XOR lemma and Impagliazzo Hardcore set

The overall goal of this approach is to use problems which we know to be only hard in the worst case, to ultimately derandomize BPP. In order to do this, an intermediate step is to amplify the hardness of the function so that it becomes hard on average.

Towards this, we consider the following: can we increase the average-case hardness of a function? The following famous lemma by Yao gives us a way to do this using the simple XOR operation.

Lemma 3.1 (Yao's XOR Lemma)**.** *For every boolean function $f : \{0,1\}^n \to \{0,1\}$, $\delta > 0$ and $k \in \mathbb{N}$, if $\epsilon > 2(1-\delta)^k$, then*

$$H^{\frac{1}{2}+\epsilon}_{avg}\left(f^{\oplus k}\right) \geq \frac{\epsilon^2}{400n} H^{1-\delta}_{avg}(f)$$

where $f^{\oplus k}(x_1, \ldots, x_k) = f(x_1) \oplus \cdots \oplus f(x_k)$.

This lemma says that computing $f^{\oplus k}$ with $1/2 + \epsilon$ probability, a much lower probability, requires nearly the same size circuit as computing f with probability $1 - \delta$, a much higher probability.

We use the following lemma to prove Yao's lemma. This lemma says that if for every circuit, there is a large set on which it makes mistakes, then we can almost "switch quantifiers" - *i.e.* we can nearly say that there is a large common set which all circuits fail.

Definition 3.2. Let $0 < \delta < 1$. A probability distribution over $\{0,1\}^n$ is said to have δ-density if for every $x \in \{0,1\}^n$, we have $\Pr[H = x] \leq \frac{1}{\delta 2^n}$.

Lemma 3.3 (Impagliazzo Hardcore Lemma)**.** *For every $\delta > 0$, boolean function $f : \{0,1\}^n \to \{0,1\}$ and $\epsilon > 0$, if $H^{1\text{-}\delta}_{avg}(f) \geq S$, then there is a δ-dense distribution H such that for every circuit C of size at most $\frac{\epsilon^2 S}{100n}$,*

$$Pr_{x\sim H}[C(x) = f(x)] \leq \frac{1}{2} + \epsilon.$$

4 Proof of Yao's XOR Lemma assuming Impagliazzo's Lemma

The main idea behind the proof can be simply stated thus:

if all S-sized circuits make errors on ($\delta 2^n$) inputs when computing f, then all (approx S)-sized circuits while computing

$$\underbrace{f \oplus f \oplus ... \oplus f}_{k \text{ times}}$$

will make errors on nearly half the inputs - (0.5 - $2(1-\delta)^k$)inputs.

One main reason is that the output of the k-fold xor will depend on all the values of f. The lemma is not true, e.g. for the k-fold OR or the k-fold AND of the values of f, since they are significantly more probable to get right.

Proof Overview

The proof shows that $\neg YXL \Rightarrow \neg IHT$.

Suppose YXL is false, and there is a circuit C which computes $f(x,y) = f(x) \oplus f(y)$ on at least $(0.5 + 2(1-\delta)^2)2^{2n}$ possible input pairs (Note the input size is now 2^{2n}.)

Let H be an arbitrary δ-density distribution.

We now show that it cannot have a "hard core".

I. Consider the following probabilistic process:

1. Toss a coin with probability of heads δ.
2. If the outcome is heads, then pick a random element according to H.
3. Otherwise pick a random element according to G, where

$$G(x) = \frac{1}{2^n} - \Pr[H = x]\frac{\delta}{1-\delta} \tag{2}$$

can be viewed as the "complement" distribution to H. Verify that G is a distribution.

Then the probability distribution of the experiment is

$$\delta H + (1-\delta)G = U_n, \tag{3}$$

the uniform distribution on n-long strings. (Verify)

II. Pick two strings independently at random according to $(U_n)^2$. By the previous experiment,

$$U_n^2 = \delta^2 HH + (1-\delta)\delta GH + \delta(1-\delta)HG + (1-\delta)^2 GG. \tag{4}$$

III. Analysis

We now show that the problem of computing $f(x), x \sim H$, can be "reduced" to the problem of computing $f(x) \oplus C(x, y)$ for some fixed y and a good circuit C. We show that this will succeed in computing $f(x)$ with sufficiently high probability, proving that H does not have a hard core. We proceed as follows.

Notation. Let $D \in \Sigma^{2n}$ be a distribution. Correspondingly, let P_D be the probability that

$$C(a, b) = f(a) \oplus f(b) \tag{5}$$

when a and b are drawn randomly according to D.

Then, we know

$$\frac{1}{2} + \varepsilon \quad \leq \quad P_{(U_n)^2} \quad \leq \quad \delta^2 P_{H^2} + (1-\delta)\delta P_{GH} + \delta(1-\delta) P_{HG} + (1-\delta)^2 P_{G^2}. \tag{6}$$

We can simplify and discard one of the above terms: $\epsilon > 2(1-\delta)^2$ and $P_G^2 < 1$, hence, we can replace $(1-\delta)^2 P_{G^2} < \epsilon/2$.[1]

Hence

$$\frac{1}{2} + \frac{\epsilon}{2} \quad \leq \quad \delta^2 P_{H^2} + (1-\delta)\delta P_{GH} + \delta(1-\delta) P_{HG}. \tag{7}$$

Since all the coefficients are less than one, at least one of the above probabilities on the right must be greater than $\frac{1}{2} + \frac{\epsilon}{2}$. (Such arguments are called *averaging arguments*.)

Let us assume, without loss of generality, that $P_{GH} > \frac{1}{2} + \frac{\epsilon}{2}$. (The other two cases will proceed in an analogous manner.) Then

$$P[C(a, b) = f(a) \oplus f(b) \mid a \sim G, b \sim H] \quad \geq \quad \frac{1}{2} + \frac{\epsilon}{2}.$$

That is,

$$\sum_{a,b} P[G = a] \quad P_{b \sim H}[C(a, b) = f(a) \oplus f(b) \mid a] \qquad \geq \frac{1}{2} + \frac{\epsilon}{2}.$$

Since we choose a and b independently, we can write this as:

$$\sum_{a,b} P[G = a] \quad P_{b \sim H}[C(a, b) = f(a) \oplus f(b)] \qquad \geq \frac{1}{2} + \frac{\epsilon}{2}.$$

By the averaging argument, there is a fixed a such that

$$P_{b \sim H}[C(a, b) = f(a) \oplus f(b)] \geq \frac{1}{2} + \frac{\epsilon}{2}. \tag{8}$$

[1] *Commentary*: The reason we discard the GG distribution is because we ultimately want to show that H is does not have a hardcore, and GG cannot do that. So we discard the GG case. This is why we define $\epsilon > 2(1-\delta)^2$ in the hypothesis in the theorem, rather than, say $\epsilon > 2\delta(1-\delta)$. Another reason for this particular choice of ϵ is the heuristic argument given before, but this is not a reason pertinent to the present proof.

We can construct a circuit $C' : \Sigma^n \to \Sigma$ by

$$C'(b) = C(a, b) \oplus f(a) \tag{9}$$

If $C(a, b) = f(a) \oplus f(a)$, then $C(b) = f(b)$. Hence,

$$P_{b\sim H}[C'(b) = f(b)] \geq \frac{1}{2} + \frac{\epsilon}{2}, \tag{10}$$

hence H does not have a hard core.

The proof generalizes to arbitrary $k \geq 2$ by induction, and using the properties of the xor function. This completes the proof of Yao's XOR lemma if we assume the Impagliazzo hardcore theorem.

5 Proof of Impagliazzo's Hardcore Lemma

Let f be a boolean function and assume that $H^{1-\delta}_{\text{avg}}(f) \geq S$. Let $\epsilon > 0$ be arbitrary. We need to construct a δ-density distribution H on which every circuit C of size $\frac{\epsilon^2 S}{100n}$ cannot compute f with probability $1/2 + \epsilon$.

We view the situation as a two-player zero-sum game. There is a player $\mathcal{D}$ which picks a δ-density distribution H, and $\mathcal{C}$ which picks a circuit to compute f. Suppose H is not a hardcore distribution for f.

By the *von Neumann MinMax theorem*, there is a mixed-strategy (i.e. randomized strategy) such that even if

1. First, $\mathcal{C}$ picks by picking an S-sized circuit according to a distribution J, and
2. Next, $\mathcal{D}$ plays a δ-dense distribution H,

it follows that H cannot be a hardcore distribution for f. That is,

$$\text{Pr}_{C\sim J, x\sim H}\left[C(x) \neq f(x)\right] \geq \frac{1}{2} + \epsilon.$$

The number of strings x such that for a circuit drawn at random according to J, we have $C(x) \neq f(x)$, is at most $\delta 2^n$.

Construct a circuit C as follows. Set $t \geq \frac{50n}{\epsilon^2}$ as a sufficiently large odd number. Randomly select t circuits according to J, denoted $C_1, \ldots, C_t$. Then for every n-length input x, $C(x)$ outputs the majority of $C_1(x), \ldots, C_t(x)$ - that is, if more than $t/2$ output 1, then $C(x) = 1$, otherwise $C(x) = 0$.

Using the Hoeffding bound (see below), we can show that for every good string x,

$$\text{Pr}_{C_1,\ldots,C_t\sim J}\left[C(x) \neq f(x)\right] < 2^{-n}.$$

(The above experiment is the following: first, pick any good string. Then *randomly* pick $C_1, \ldots, C_t$ according to the distribution J. We then consider the probability of correctly computing f.)

Since there are at most 2^n good strings, by taking the union bound over all possible choices of circuits, we conclude that the probability that every set of t circuits is bad is at most $2^n \times 2^{-n}$, which is strictly less than 1. This means that there is at least one C is correct on every good string x.

This in turn implies that the circuit C, which has size at most $tS = \frac{50n}{\epsilon^2}$ can compute correctly on at least $2^n - \delta 2^n = (1 - \delta)2^n$ strings. Hence $H^{1-\delta}_{\text{avg}}(f) < S$, which contradicts our assumption.

Hoeffding Bound

Lemma 5.1 (Hoeffding, 1962)**.** *If $X_1, \ldots, X_t$ are independent, identically distributed random variables, define*

$$\overline{X} = \frac{\sum_{i=1}^{t} X_i}{t} \quad \mu = \mathbb{E}[X],$$

then we have

$$Pr\left[|\overline{X} - \mu| \geq \epsilon\right] \leq 2e^{-2t\epsilon^2}.$$

We apply the lemma as follows to get the bound in the proof of Impagliazzo Hardcore lemma.

Let x be a string with $f(x) = 1$. Let X_i be a random variable which is 1 if $C_i = 1$ and 0 otherwise. Then the majority of $X_1, \ldots, X_t$ is 0 (*i.e.* our output is *wrong*) if and only if $\frac{\sum_{i=1}^{n} X_i}{t} < 1/2$.

The expected value, $\mathbb{E}[X_1]$ is $\geq \frac{1}{2} + \epsilon$.

Thus, the majority is wrong if and only if $\frac{\sum_{i=1}^{n} X_i}{t} - \mathbb{E}[X_1] > \epsilon$. By Hoeffding bound, this event occurs with probability at most $2e^{-2t\epsilon^2} < 2^{-n}$.

The case when $f(x) = 0$ is similar.

6 Points to Ponder

1. Where exactly did we need circuits instead of algorithms? [Hardcoding $f(a)$, for one place.]
2. Where did the assumption that H is δ-dense play a crucial part? [One place: when we verify that G is a distribution, we need the assumption that $H(x) \leq \delta 2^{-n}$ for G to be non-negative.]