CS760 Topics in Computational Complexity **2024-25 Sem I**

Lecture 16: 01-10-2024

Scribe: Atharva Bendale, Tanmay Patil *Lecturer: Rohit Gurjar*

In this lecture we will take a look at the **Switching Lemma** which helps us prove the following theorem:

Theorem 16.1. *PARITY is not in AC^0.*

We will show a stronger result that for sufficiently large n, and any constant $k \geq 2$, any depth-k (unbounded fan-in) circuit for parity function on n bits require size $2^{\Omega(n^{1/(k-1)})}$.

1 random restriction

The idea of random restriction is to randomly choose some input bits of a circuit and fix them to some random values. The hope is that after such random restriction, the circuit will becomes something simpler and easier to analyze. For example, in Hastad's switching lemma, which we will see later, after hitting a random restriction to a DNF or CNF, the resulting (restricted) function turns out to have a small depth decision tree with high probability.

We formally define a random restriction for any boolean function f as follows:
Let $f : \{0,1\}^n \to \{0,1\}$ be a n-bit boolean function, we take $0 \leq \sigma \leq 1$ parameterizing the random restriction as follows:

1. Randomly choose a subset of size $\sigma \cdot n$ from the set of all boolean variables, these are called $\star$ variables (star variables).
2. We randomly set all variables other than the $\star$ variables to 0 or 1 with probability 1/2.

We denote this restriction as ρ, the set of $\star$ variables as $stars\,(\rho)$ and the resulting restricted function as $f|_\rho$. We prove Theorem 16.1 by using the result that random restriction applied on PARITY still results in PARITY or its negation and by showing that applying random restrictions multiple times on an AC^0 circuit converts it to a depth-2 circuit i.e. a CNF or DNF. And we know that any CNF or DNF requires exponential size to represent the parity function. We introduce the switching Lemma for proving this theorem.

2 Random restrictions on AC^0

Width of a clause is the number of literals in it. For a CNF or a DNF, its width is the maximum width of any clause in it. The high level idea is to show that we can replace any small width AND of ORs with a small width OR of ANDs by applying random restrictions. This gives us a way to reduce the depth of an AC^0 circuit.

We are going to make some assumptions on the AC^0 circuit, which are without loss of generality:

1. The bottommost gates (which are OR/AND of some input gates) should all have small width. By small width we mean that any of the gates in the bottom layer should not have more than $O\,(log\ n)$ inputs. We will argue that this can be achieved by a random restriction, which kills all larger width nodes with high probability.
2. All not gates are at the input level. We can push down any intermediate NOT gates below to achieve this.
3. We have alternating layers of AND and OR gates. This can be achieved while circuit size remaining polynomially bounded.

Let's consider that the bottom-most layer of the AC^0 circuit under consideration is an AND layer and that the depth of the circuit is d. Let the bottom most layer (AND layer) be layer 1, the layer above it layer 2 and so on.
We now show that any random restriction which sets significant fraction of variables will kill nodes having more than $O(log\ n)$ inputs with high probability.
Probability p that any parent node with k input nodes will survive after a random restriction, assuming that $1-\rho$ fraction of input nodes are set on an average for each parent node is:

$$p = \big(1/2\big)^{(1-\rho)\cdot k}$$

For any such parent node with $k > O(log\ n)$ and sufficiently large n:

$$p < \big(1/2\big)^{(1-\rho)\cdot c(log_2\ n)}$$

$$p < \big(1/n\big)^{c(1-\rho)}$$

Therefore any nodes having more than $O\left(log\ n\right)$ nodes will be killed with high probability. This is only a rough argument. One has to use Chernoff bounds to show that for any gate with large enough width, at least one input will be set to zero, with high probability.

After the first random restriction, all the nodes in layer 1 have small width with high probability as argued above. As we know that layer 2 and layer 1 together form a DNF like structure, we want to convert this to CNF, using random restrictions. Note that without any restriction, there is trivial way of converting a DNF into a CNF, but that can result in exponential size. With random restrictions, the conversion to CNF keeps the width small, and hence the size is also small. The conversion goes via decision trees, which we define below.

3 Decision Tree

Decision tree (DT) is a computational model used for representing query-based algorithms (or functions). For boolean function, a decision tree is simply a binary tree with internal vertices associated by a variable x_i and the two out-going edges correspond to x_i being set to 0 or 1. The leaves of a decision tree are indexed by 0 or 1. The boolean function computed by a decision tree T is naturally defined as starting from the root and going downward according to the value of the variables and finally outputting the value in the leaf one ends at. See the following for an example:

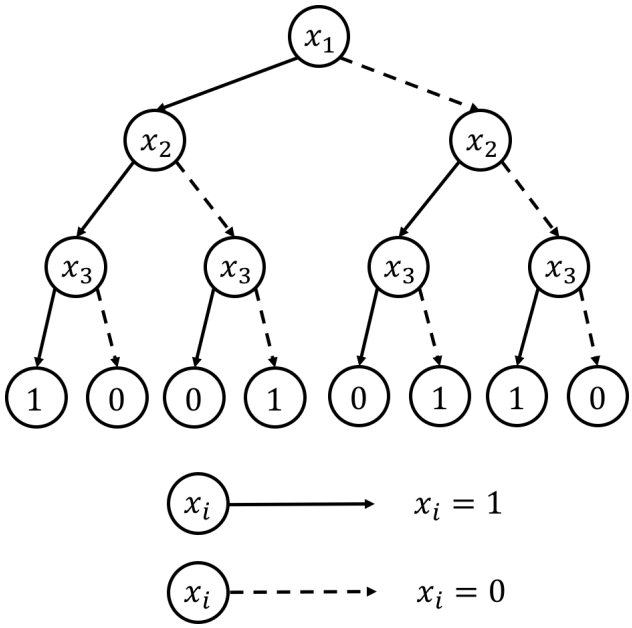

Figure 1: The decision tree representing the parity function on three variables.

Depth of a decision tree is the maximum of depth over all of the branches. Optimal decision trees for any boolean function are trees having minimum depth. We represent depth of the optimal decision tree for any boolean function f by $DT_{depth}(f)$. It is a fact that $DT_{depth}(PARITY_n) = n$ as the correct output can only be determined after looking at all of the n variables and therefore any decision tree solving $PARITY_n$ will need to be of depth n. The following lemma relates DNF/CNF representation of a Boolean function with decision tree representation. Recall that width of a DNF/CNF is the maximum number of literals in any of its clauses.

Lemma 16.2. *If for any boolean function f, $DT_{depth}(f) \leq d$, then f has a width d DNF and also a width d CNF.*

We can construct a DNF by simply taking OR of all the paths in the decision tree starting from root and leading to output 1. For example consider the above Figure 1, the obtained DNF formula should be:

$$(x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (\neg x_1 \wedge x_2 \wedge \neg x_3) \vee (\neg x_1 \wedge \neg x_2 \wedge x_3)$$

Similarly a CNF can be obtained where each of its clause is the negation of the term describing a path from the root to a leaf labeled with 0. For example consider the above Figure 1, the obtained CNF formula should be:

$$(\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_3)$$

4 Switching Lemma

Let f be a function with a DNF formula of width at most w over n variables. Let α be a random restriction with $s = \sigma n$ stars, where $\sigma \leq 1/5$. Then for each $d \geq 0$ (and $\leq s$),

$$Pr[DT_{depth}(f|_\alpha > d)] \leq (10\sigma w)^d.$$

5 Proof of Switiching Lemma

To prove the above lemma, we consider two sets of restrictions :

$$\mathcal{R}_s : \text{set of all restrictions with s stars}$$

$$\mathcal{B} : \text{set of all the bad restrictions } \beta \text{ (making } DT_{depth}(f|_\beta) > d)$$

Take f to be a DNF formula, $f = T_1 \vee T_2 \vee T_3 \vee \cdots$. First, we begin by defining a one to one map (encoding) from $\mathcal{B}$ to $\mathcal{R}_{s-d} \times$ (a small auxiliary info set).

5.1 Encoding

The encoding maps $\mathcal{B}$ to $\mathcal{R}_{s-d}$ with some auxiliary information set required. The information required can be thought of as the information required to get back a restriction $\beta \in \mathcal{B}$ from its respective element in $\mathcal{R}_{s-d}$. Note that $\mathcal{R}_{s-d}$ is a smaller set than $\mathcal{R}_s$. A rough estimate gives us

$$\frac{|\mathcal{R}_{s-d}|}{|\mathcal{R}_s|} \approx (2\sigma)^d.$$

The encoding should somehow take into account the fact that β is a bad restriction.

How to go about with the process of encoding. Let's consider

$$f|_\beta = T_{i_1} \vee T_{i_2} \vee T_{i_3} \cdots$$

where T_{i_1} is the first surviving clause under the restriction β and so on (All other clauses were killed because they were set to zero). Let T_j be the clause of f which became T_{i_1} under restriction β. Now if U_1 is the set of all the alive variables in T_{i_1}, then all the other literals not in U_1 but in the clause T_j were set to one. Now let's take a systematic approach to find the encoding:
Step 1: First let us set all variables in U_1 such as to make T_{i_1} one. Let γ_1 be the assignment of U_1 which ensures that T_{i_1} is one. The idea behind this step is that T_{i_1} can be recovered from the restriction $\beta\gamma_1$ on f. Why? T_{i_1} can be recovered because it is the first term in f which becomes 1 under $\beta\gamma_1$. Note that we don't know which part of $\beta\gamma_1$ is coming from β and which is coming from γ_1. To recover that we will store some auxiliary information about variables set by assignment γ_1. If we do this naively, then we will need $|U_1| \log n$ bits. However, this multiplies a factor of $n^{|U_1|}$ in the size of auxiliary info set. Overall when we set d variables, we get a factor of n^d, and this gives us a meaningless bound on $\mathcal{B}$.

Instead, a clever way to store the set U_1 is its corresponding set of indices in T_j. Since T_j has only w literals, U_1 can be represented in form of auxiliary information with atmost $|U_1| \log(w)$ bits.

Step 2: While γ_1 assignment helps us in recovering β, the function just becomes constant on assignment γ_1. Hence, it is not clear how to proceed further after assignment γ_1. To proceed further, we choose a different assignment π_1 of U_1 as follows. Recall that the $DT_{depth}(f|_\beta) > d$). Consider a decision tree for $f|_\beta$ which first queries the variables in U_1. There must be a path in this tree which has depth $> d$. Let π_1 be the assignment to U_1 variables obtained by following this path. As there is only one assignment γ_1 which can make T_{i_1} one, π_1 will surely make T_{i_1} to be zero.

In the next round, we will work with $f|_{\beta\pi_1}$. Hence, for decoding we will also need to know π_1. Now any π_1 can be represented simply by $|U_1|$ bits (what was each variable in U_1 assigned). Thus this auxiliary information in the form of $|U_1|$ bits is also required. We can also say that

$$DT_{depth}(f|_{\beta\pi_1}) > d - |U_1|$$

because the depth of decision tree under β was $> d$ and now we just assigned U_1 variables along a depth $> d$ path in the tree, thus reducing the depth of the tree by $|U_1|$.

To summarize the two steps, after the first round our encoding for β will be $\beta\gamma_1$ together with $|U_1| \log w + |U_1|$ bits storing variables of U_1 and the assignments under π_1.

Step 3: Repeat step 1 and step 2 for the first clause that survives in $f|_{\beta\pi_1}$. Continue this process for the first surviving clause in $f|_{\beta\pi_1\pi_2}$ and so on. Stop the process when d variables are assigned.

5.2 Calculations

We got $\text{Enc}(\beta) = \beta\gamma_1\gamma_2\cdots\gamma_l +$ some auxiliary info, where $\beta\gamma_1\gamma_2\cdots\gamma_l$ is in $\mathcal{R}_{s-d}$.
Now let's calculate the cardinality of $\mathcal{B}$. The auxiliary info set can always be represented by bits. Thus, we need to find the total number of auxiliary info bits required for the encoding scheme given above.
Let $|U_i| = d_i$, thus from step 3, we immediately have $\sum d_i = d$. So for decoder to determine the γ_i and π_i, it requires at most $d_i log(w) + d_i$ bits of additional info. But the decoder also does not how much d_i actually is, and so we require one more sentinel symbol along with w symbols. So the decoder actually requires $d_i \log(w+1) + d_i$ bits for T_{j_i}. Total number of additional info bits required are

$$\sum d_i \log(w+1) + d_i = d\log(w+1) + d \leq d\log(w) + 2d \text{ (As } w \geq 1)$$

Thus we have shown an injective mapping from $\mathcal{B}$ of bad restrictions into

$$\mathcal{R}_{s-d} \times \{0,1\}^{d\log(w)+2d}$$

This set has cardinality

$$|\mathcal{B}| = \binom{n}{s-d} 2^{n-(s-d)}(4w)^d.$$

Since the cardinality of all s-star restrictions $\mathcal{R}_s$is

$$|\mathcal{R}_s| = \binom{n}{s} 2^{n-s},$$

we conclude the probability of getting a bad restriction is

$$Pr[DT_{depth}(f|_\beta > d)] = \frac{|\mathcal{B}|}{|\mathcal{R}_s|}$$

which is at most

$$\begin{aligned}\frac{\binom{n}{s-d}2^{n-(s-d)}(4w)^d}{\binom{n}{s}2^{n-s}} &= \frac{s(s-1)(s-2)\cdots(s-d+1)}{(n-s+d)(n-s+d-1)\cdots(n-s+1)}(8w)^d \\ &\leq \left(\frac{s}{n-s}\right)^d (8w)^d \leq \left(\frac{\sigma}{1-\sigma}\right)^d (8w)^d \leq (10\sigma w)^d.\end{aligned}$$

where we used $\sigma \leq \frac{1}{5}$ in the last step.