# Contents

| CSC2429 / MAT1304 | February 21, 2019 |
|---|---|

# Lecture : Circuit Complexity

*Instructor: Benjamin Rossman* — *Scribe: Lily Li*

## 1 Administrivia

- **Instructor:** Ben Rossman.
- **Course Info:** Available at the course website. Just in case the website is down: lectures are Thursdays from 16:00 to 18:00 in Bahen B026. Office hours are by appointment.
- **Textbook:** *Boolean Function Complexity* by Stasys Junkha. This is available as a free eBook through the University of Toronto library.
- **Prerequisites:** None, but a previous complexity course is useful. Please read Appendix A.1 of the textbook and understand the material.
- **Workload:** Homework assignment(s), scribe notes, paper report (5 to 10 pages), and presentation if you so choose. No exams.

## 2 Basic Definitions

### 2.1 Boolean Functions

**Definition 1.** *A $n$-**ary Boolean function** $f$ is a function of the form $f : \{0,1\}^n \to \{0,1\}$. Usually we interpret $(0,1)$ as $(\mathsf{FALSE}, \mathsf{TRUE})$ or as $(1,-1)$ — this makes sense if you think of it as $(-1)^0$ and $(-1)^1$.*

Let $\{0,1\}^* = \cup_{n \in \mathbb{N}} \{0,1\}^n$. We typically refer to a family of Boolean function(s) $f : \{0,1\}^* \to \{0,1\}$. This corresponds to a sequence of functions $f_n : \{0,1\}^n \to \{0,1\}$ and to a language $L \subseteq \{0,1\}^*$ described by its characteristic function $f_L : \{0,1\}^* \to \{0,1\}$.

**Example 2.** *The following are some examples of $n$-ary Boolean functions:*

1. $PARITY(x_1, ..., x_n) = \sum_{i=1}^n x_i \mod 2$.
2. $MOD_p(x_1, ..., x_n) = 1 \iff \sum_{i=1}^n x_i \equiv 0 \mod p$.
3. $MAJORITY_n(x_1, ..., x_n) = 1 \iff \sum_{i=1}^n x_i \geq \lceil n/2 \rceil$.
4. $k - CLIQUE : \{0,1\}^{\binom{n}{2}} \to \{0,1\}$. *Think of each graph $G$ as an indicator vector $\mathbb{1}_G$ over its $\binom{n}{2}$ edges. Then $k - CLIQUE(\mathbb{1}_G) = 1$ if and only if $G$ has a $k$-clique.*

Let us consider DeMorgan circuits. These contain logical connectives $\{\vee, \wedge, \neg\}$, input variables $\{x_1, ..., x_n\}$, and constants $\{0, 1\}$.

**Definition 3.** *A $n$-**ary DeMorgan circuit** is a finite directed acyclic graph (DAG) with nodes labelled as follows:*

- *Nodes of in-degree zero ("inputs") are labelled by a variable or a constant.*
- *Non-input nodes ("gates") of in-degree one are labelled with $\neg$. Gates of in-degree two are labelled with $\vee$ or $\wedge$.*
- *A subset of the nodes are designated as "outputs" (default:* the *node with out-degree zero).*

*Two circuits are **equivalent** if they compute the same function.*

**Formulas** are tree-like circuits. Since different branches in a formula depend on different copies of the variables, formulas are memory-less. See Figure 1. Proving that formulas are polynomially weaker than circuits is still an open problem.

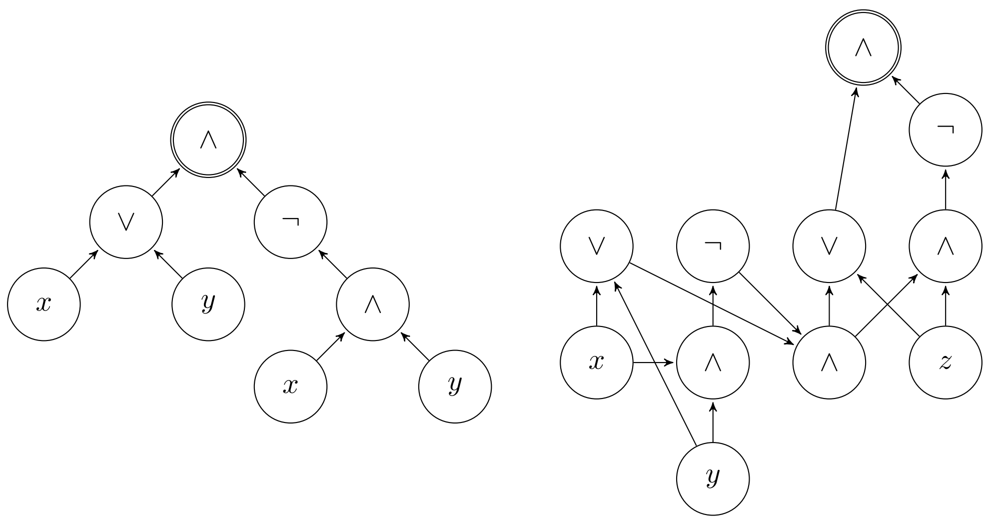


Figure 1: (Left) Formula computing $x \oplus y$. (Right) Circuit computing $x \oplus y \oplus z$.

**Definition 4.** *The **size** of a circuit is the number of $\vee$ and $\wedge$ gates it contains.*

*The **leaf-size** of a formula is the number of leaves in its associated DAG. This is one more than the circuit size as defined above.*

*The **circuit size** of an $n$-ary Boolean function $f : \{0,1\}^n \to \{0,1\}$, written $\mathcal{C}(f)$, is the minimum size of a circuit computing $f$. Similarly, the **formula (leaf) size** of $f$, written $\mathcal{L}(f)$, is the minimum size of a formula computing $f$.*

*The **depth** of a circuit is the maximum number of $\wedge$ and $\vee$ gates on any input to output path.*

**Example 5.** *It is a major open problem to compute the circuit and leaf size lower bounds for various Boolean functions. A couple of known results are as follows.*

| $f$ | $\mathcal{L}(f)$ | $\mathcal{C}(f)$ |
|---|---|---|
| $AND_n$ | $n$ | $n-1$ |
| $PARITY_n$ | $\Theta\left(n^2\right)$ | $3(n-1)$ |

*The results for $AND_n$ are tight since the output depends on all the inputs. Improving the gap size between $\mathcal{L}(PARITY_n)$ and $\mathcal{C}(PARITY_n)$ would separate $\mathsf{NC}_1$ from $\mathsf{P}$.*

### 2.2 Other Ways of Measuring Size

Other ways of counting the size of a circuit include: (1) counting the number of wires and (2) counting all gate types (including $\neg$ gates). It turns out that the result of these calculations differ from our definition above by at most a factor of two. It should be easy to see why this is in the former case. Every $\wedge$ and $\vee$ gate has two incoming wires. Claim 7 shows this in the latter case.

**Definition 6.** *The input to every $\neg$ gate in a circuit in* ***negation normal form*** *is a variable. See Figure 2.*

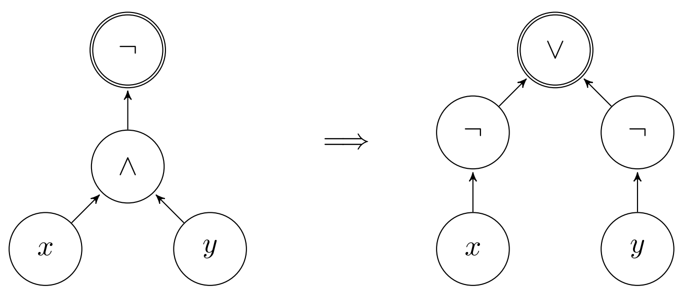


Figure 2: Apply DeMorgan's Law to all $\neg$ gates whose inputs are not literals on the left circuit to get the equivalent right circuit in negation normal form.

**Claim 7.** *Every circuit $C$ of size $m$ is equivalent to a circuit in negation normal form of size $\leq 2m$.*

*Proof.* Apply DeMorgan's law to every $\neg$ gate whose input is not a variable. This switches the order of $\neg$ and $\wedge/\vee$ in the DAG and adds an additional $\neg$ gate. By the end of the process we have added at most $m$ $\neg$ gates. □

Thus we can push all $\neg$ gates to the bottom and interpret the inputs as literals (variables and their negation). We can also modify the definition of leaf-size to only count leaves leading to literals (never-mind the constants).

### 2.3 General Basis

A **basis** $B$ is a set of Boolean functions (or "gate types"). Examples of basis include:

- DeMorgan basis: $\{\wedge, \vee, \neg\}$.

- Full binary basis: all Boolean functions $\{0,1\}^2 \to \{0,1\}$ (for example, you would get $\oplus$).
- Monotone basis: $\{\wedge, \vee\}$ (NOT universal).
- $\mathsf{AC}^0$ basis: $\{\wedge^k, \vee^k, \neg : k \in \mathbb{N}\}$ which are unbounded fan-in $\wedge$ and $\vee$ gates.

For a function $f$, let $\mathcal{L}_B(f)$ and $\mathcal{C}_B(f)$ be the leaf and circuit size of $f$ with formulas and circuits built from gates of basis $B$. A basis is **universal** if it computes all functions. For two universal basis $B_1$ and $B_2$ it is possible to build a circuit using gates from $B_1$ which simulates any gate from $B_2$. If all functions in $B_1$ and $B_2$ have constant arity, it follows that $\mathcal{C}_{B_2}(f) = O(\mathcal{C}_{B_1}(f))$; for formula size, the relation is $\mathcal{L}_{B_2}(f) = \mathcal{L}_{B_1}(f)^{O(1)}$. This polynomial blow-up is unavoidable in some cases. Recall the function $PARITY_n$: $\mathcal{L}_{\{\wedge,\vee,\neg\}}(PARITY_n) = \Theta(n^2)$ whereas $\mathcal{L}_{\{\oplus\}}(PARITY_n) = n-1$.

## 2.4 Models of Computation

**Definition 8.** *A **uniform model of computation** is a single machine/program with a finite description which operates on all inputs in $\{0,1\}^*$. Examples range from simple finite automata (where we have lower bounds ala the pumping lemma) to complex Turing Machines (lower bounds much harder to come by).*

*Recall that a language $L \subseteq \{0,1\}^*$ can be interpreted as a sequence of functions $(f_0, f_1, ...)$ where $f_n : \{0,1\}^n \to \{0,1\}$ and $f_n(\mathbf{x}) = 1 \iff \mathbf{x} \in L$ for any $\mathbf{x} \in \{0,1\}^n$. A **non-uniform (concrete) model of computation** is a sequence $(C_0, C_1, ...)$ of combinatorial objects (namely circuits) where $C_n$ computes $f_n$. Examples include: circuits in the DeMorgan basis, restricted class of circuits (formulas, monotone model), decision trees, etc.*

Observe that the non-uniform model of computation is more powerful than the uniform one since the finite program can be used as every combinatorial objects in the sequence. It follows that lower bounds in the non-uniform model also imply lower bounds in the uniform model. While upper bounds in the uniform model imply upper bounds in the non-uniform model. *We want: unconditional lower bounds.*

Circuits efficiently simulate Turing Machines.

**Lemma 9.** *Any Turing Machine (TM) $M$ with running time $t(n)$ can be simulated by a circuit (family of) of size $O\left(t(n)^2\right)$.*

Exercise for the reader. *Hint: think about configurations of the Turing Machines as a $t(n) \times t(n)$ grid and construct a circuit for every grid cell.* Fischer and Pipenger (1979) proved an $O(t(n) \log t(n))$ upper bound on *oblivious Turing Machines*[1]. It is unknown if we can do better.

**Corollary 10.** *If there is a super polynomial lower-bound (better than $\Omega(n^c)$ for all constants $c > 0$) on the circuit size of any language in $\mathsf{NP}$, then $\mathsf{P} \neq \mathsf{NP}$.*

Finding the lower-bound would actually show $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ where $\mathsf{P/poly}$ is the class of languages decidable by poly$(n)$-size circuits.

[1] An oblivious TM is one whose head motion depends only on the size of the input and not its particular bits. Take a look at this blog post for some entertainment.

We will see some polynomial lower bounds for formulas in the DeMorgan basis later on. As a historical curio, the following is a catalogue of lower bound results for an explicit Boolean function:

1. $\Omega(n^{1.5})$ Subboboskay '61
2. $\Omega(n^2)$ Khrapchenko '71
3. $\Omega(n^{2.5-o(1)})$ Andreev '83
4. $\Omega(n^{3-o(1)})$ Håstad '98 (this is the state of the art until very recently).

# 3 DeMorgan Basis

## 3.1 Balancing Formulas

Next we consider the relationship between circuit size and depth. First observe that every circuit of depth $d$ is equivalent to a formula of size at most $2^d$. To see this, take the circuit and duplicate any branches that gets reused. The resulting binary tree has at most as many nodes as a perfect binary tree of depth $d$ which itself has circuit size $2^d$.

The next theorem shows the converse: every formula of size $s$ can be "balanced" to obtain a formula of depth $O(\log s)$.

**Theorem 11.** ***(Spira 1971).*** *Every formula with leaf-size $s$ is equivalent to a formula of depth $O(\log s)$ ($2\log_{3/2}(s)$ to be exact) and thus size at most $s^{O(1)}$ ($s^{2/\log_2(3/2)}$).*

*Proof.* By induction on $s$. The base case is trivial. Let $F$ be the original formula and $g$ be some gate. Let $F_g$ be the sub-formula rooted at $g$. For $b \in \{0,1\}$, let $F^{(g\leftarrow b)}$ be the formula with $g$ replaced with the constant value $b$. See Figure 3. Note that $\mathcal{L}(F) = \mathcal{L}(F_g) + \mathcal{L}\left(F^{(g\leftarrow b)}\right)$. Minimize $\mathcal{L}(F)$ by balancing the two terms on the RHS. By Claim 12, we can find a gate $g$ such that $\frac{s}{3} \leq \mathcal{L}(F_g) \leq \frac{2s}{3}$.

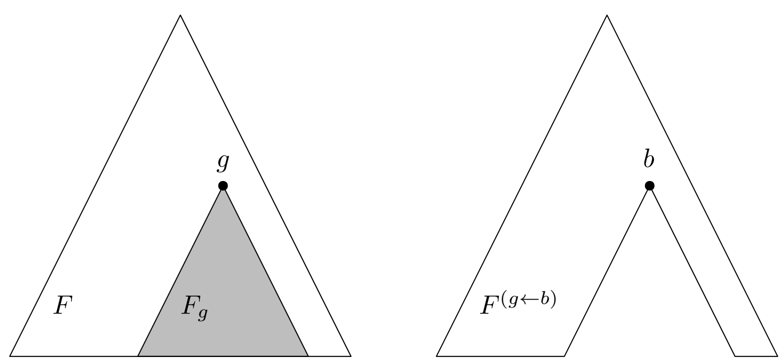


Figure 3: Illustration of gate $g$ and formulas $F_g$ and $F^{g\leftarrow b}$.

Note that $F \equiv \left(F_g \wedge F^{(g\leftarrow 1)}\right) \vee \left((\neg F_g) \wedge F^{(g\leftarrow 0)}\right)$; $F_g$ must evaluate to 0 or 1 and the formula does just that. Apply the induction hypothesis to the four formulas $F_g$, $F^{(g\leftarrow 1)}$, $\neg F_g$, and $F^{(g\leftarrow 0)}$ to get formulas of depth $\leq 2\log_{3/2}(2s/3)$. The original formula $F$ can only grow by at most depth two so

$$\begin{aligned}
\mathrm{depth}(F) &\leq \max\left\{\mathrm{depth}\left(F_g\right), \mathrm{depth}\left(F^{(g\leftarrow 1)}\right), \mathrm{depth}\left(\neg F_g\right), \mathrm{depth}\left(F^{(g\leftarrow 0)}\right)\right\} + 2 \\
&\leq 2\log_{3/2}\frac{2s}{3} + 2 \\
&= (2\log_{3/2} s - 2) + 2 \\
&= 2\log_{3/2} s
\end{aligned}$$

Thus there exists a formula equivalent to $F$ of depth at most $O(\log s)$. $\square$

**Claim 12.** *There exists a gate $g$ such that $F_g$ has leaf-size between $\frac{s}{3}$ and $\frac{2s}{3}$ leaves.*

*Proof.* Let $r \rightsquigarrow \ell$ be a root to leaf path in the DAG containing the most $\wedge/\vee$ gates. At the root $r$, $\mathcal{L}(F_r) = \mathcal{L}(F) = s$ and at the leaf $\ell$, $\mathcal{L}(F_\ell) = 1$. Starting at $r$ and moving down to $\ell$, the successive leaf-sizes can at most halve after each step. Thus there must exists a gate $g$ for which $\frac{s}{3} \leq \mathcal{L}(F_g) \leq \frac{2s}{3}$. $\square$

## 3.2 Circuit Size of General Boolean Functions

### 3.2.1 Upper Bound

Given a function $f : \{0,1\}^n \to \{0,1\}$, let us consider some upper bounds for $\mathcal{C}(f)$.

1. Brute force DNF: $O(n2^n)$. There are $2^n$ rows in the truth table of $f$. Each row specifies the output given the $n$ inputs. Thus a clause with $n-1$ $\wedge$ gates represents each row of the table. Formally we consider the expression

$$f(\mathbf{x}) = \bigvee_{\mathbf{a}\in f^{-1}(1)} (\mathbf{x} = \mathbf{a}) = \bigvee_{\mathbf{a}\in\in f^{-1}(1)} (l_1 \wedge l_2 \wedge \cdots \wedge l_{n-1} \wedge l_n)$$

   where $l_i = x_i$ if $a_i = 1$ and $l_i = \overline{x}_i$ otherwise.

2. Function decomposition: $O(2^n)$. Observe that

$$f(\mathbf{x}) \equiv (x_n \wedge f_1(\mathbf{x})) \vee (\overline{x}_n \wedge f_0(\mathbf{x}))\,.$$

   where $f_1 = f(x_1, ..., x_{n-1}, 1)$ and $f_0 = f(x_1, ..., x_{n-1}, 0)$. Thus

$$\mathcal{C}(f) \leq \mathcal{C}(f_1) + \mathcal{C}(f_0) + 3.$$

   Apply the decomposition recursively to $f_1$ and $f_0$. Generally at step $k$,

$$\mathcal{C}(f) \leq \sum_{\mathbf{a}\in\{0,1\}^k} \mathcal{C}(f_{\mathbf{a}}) + 3(2^k - 1)$$

   where $f_{\mathbf{a}}(\mathbf{x}) = f(x_1, ..., x_{n-k}, a_1, ..., a_k)$. Since $f(\mathbf{a})$ is a constant at the $n^{\text{th}}$ step, $3(2^n - 1)$ is an upper bound on the circuit size of $f$.

3. Computation reuse: $O(2^n/n)$. See Theorem 14 below.

Let $ALL_n : \{0,1\}^n \to \{0,1\}^{2^{2^n}}$ be the function which calculates all the $n$-ary Boolean functions at the same time[2]. That is $(ALL_n(\mathbf{x}))_f := f(\mathbf{x})$ for any $n$-ary Boolean function $f$.

**Claim 13.** $\mathcal{C}(ALL_n) \le O(2^{2^n})$.

*Proof.* Similar to the function decomposition analysis. For every function $f$ in the output of $ALL_n$, $f(\mathbf{x}) \equiv (x_n \wedge f_1(\mathbf{x})) \vee (\overline{x}_n \wedge f_0(\mathbf{x}))$ where $f_1 = f(x_1, ..., x_{n-1}, 1)$ and $f_0 = f(x_1, ..., x_{n-1}, 0)$. Note that $f_1$ and $f_0$ are outputs of $ALL_{n-1}$. See Figure 4. Since $ALL_n$ has $2^{2^n}$ outputs,

$$\mathcal{C}(ALL_n) \le \mathcal{C}(ALL_{n-1}) + 3\left(2^{2^n}\right) = c\left(2^{2^{n-1}}\right) + 3\left(2^{2^n}\right) \in O\left(2^{2^n}\right)$$

for some constant $c$. ☐

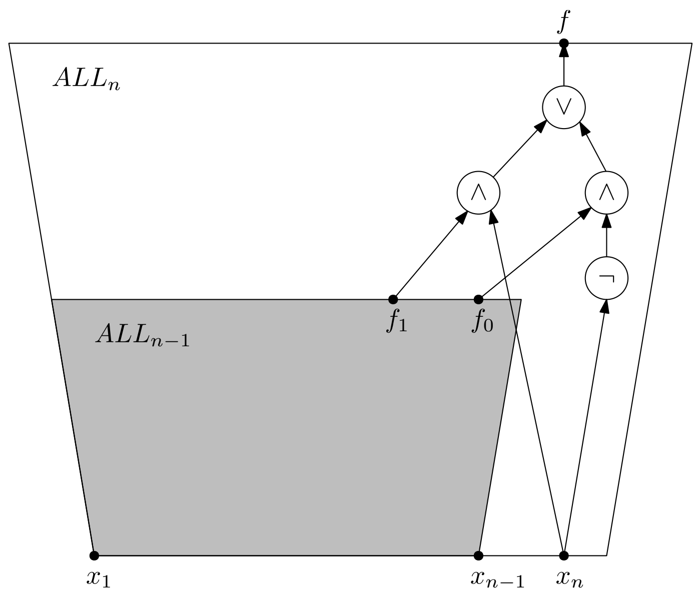


Figure 4: Obtaining a circuit for $ALL_n$ from a circuit for $ALL_{n-1}$.

**Theorem 14.** ***(Lupanov 1958).*** *Every $n$-ary Boolean function has circuit size $O(2^n/n)$*

*Proof.* The key idea is to use $ALL_{n-k}$ in place of $\{f_{\mathbf{a}} : \mathbf{a} \in \{0,1\}^k\}$ in the analysis of function decomposition. Formally, we have

$$\mathcal{C}(f) \le \sum_{\mathbf{a}\in\{0,1\}^k} \mathcal{C}(f_{\mathbf{a}}) + 3(2^k - 1) \le \mathcal{C}(ALL_{n-k}) + 3(2^k - 1) \le O\left(2^{2^{n-k}}\right) + O(2^k)$$

[2]To see that the range of $ALL_n$ is indeed $2^{2^n}$, recall that the domain of every $n$-ary Boolean function is $2^n$. There is a bijection between the set of functions and the power set of $\{0,1\}^n$ (of size $2^{2^n}$).

where the last inequality follows from Claim 13. Observe that the two terms on the RHS are balanced when $k = n - \log(n - \log n)$ since

$$\begin{aligned} O\left(2^{2^{n-k}}\right) + O\left(2^k\right) &= O\left(2^{2^{\log(n-\log n)}}\right) + O\left(2^{n-\log(n-\log n)}\right) \\ &= O\left(2^{n-\log n}\right) \\ &= O\left(2^n/n\right) \end{aligned}$$

It follows that the circuit complexity of all $n$-ary Boolean function is bounded above by $O(2^n/n)$. $\square$

### 3.2.2 Lower Bound

Prior to Lupanov's result above, Shannon showed a matching lower bound.

**Theorem 15.** ***(Shannon 1949).*** *Almost all $n$-ary Boolean functions (as $n \to \infty$) have circuit size $O(2^n/n)$.*

*Proof.* Use the counting argument. Recall the number of $n$-ary Boolean functions is $2^{2^n}$ and let $s = \frac{2^n}{n}$. We will show that the number of Boolean functions which can be computed by circuits of size $s$ is $\ll 2^{2^n}$. Let $A$ be the set of all $n$-ary circuits with $2n$ literals, $x_1, ..., x_n, \overline{x}_1, ..., \overline{x}_n$, and $s$ gates, denoted $g_1, ..., g_s$. We obtain an upper bound on the number of circuits in $A$ as follows. Each circuit can use any subset of the $s$ gates. Each $\wedge/\vee$ gate can pick two inputs from the $2n$ literals and $s-1$ other gates. If $n$ is sufficiently large (say $n \geq 100$), then $s + 2n < 3s$ so

$$|A| \leq 2^s(s+2n)^{2s} \leq 18^s s^{2s}.$$

Observe that every $n$-ary function with $\mathcal{C}(f) \leq s$ is computed by at least $s!$ distinct circuits in $A$ since we can permute the labels on the $s$ gates. Thus the total number of Boolean functions computed by circuits in $A$ is at most $\frac{|A|}{s!}$. Recall that $s! \geq \left(\frac{s}{e}\right)^s$. For $s = \frac{2^n}{n}$,

$$\frac{|A|}{s!} \leq \frac{18^s s^{2s}}{(s/e)^s} \leq 50^s s^s = 50^{2^n/n}\left(\frac{2^n}{n}\right)^{2^n/n} = \left(\frac{50}{n}\right)^{2^n/n}(2^n)^{2^n/n} \leq 2^{2^n - 2^n/n}$$

since $n \geq 100$. Thus at least $2^s$ Boolean formulas have circuit size greater than $s$. $\square$

## 3.3 Circuit Size Hierarchy

**Theorem 16.** *If $n \leq s(n) \leq \frac{2^{n-2}}{n}$, then* $\mathsf{SIZE}[s] \subsetneq \mathsf{SIZE}[4s]$.

*Proof.* Use a combination of Shannon (Theorem 15) and Lupanov (Theorem 14). Pick[3] an $m < n$ such that

$$s(n) \leq \frac{2^m}{m} \leq 2s(n).$$

[3]Such an $m$ must exists. When $m = 1$, $2^m/m \leq s(n)$ and when $m = n-1$, $2^m/m \geq s(n)$ so there must be some $m$ such that $2^m/m \leq s(n)$ and $2^{m+1}/(m+1) \geq s(n)$. If $2^{m+1}/(m+1) \geq 2 \cdot s(n)$ then

$$s(n) \leq \frac{2^m}{m+1} \leq \frac{2^m}{m}$$

which contradicts our original choice of $m$.

By Shannon, there exists a function $f : \{0,1\}^m \to \{0,1\}$ such that

$$\mathcal{C}(f) > \frac{2^m}{m} \geq s(n).$$

Thus $f \notin \mathsf{SIZE}[s]$. By the tight bound from Lupanov's theorem, $\mathcal{C}(f) \leq 2^m/m + o(2^m/m)$ so

$$\mathcal{C}(f) \leq \frac{2 \cdot 2^m}{m} \leq 4s(n)$$

and $f \in \mathsf{SIZE}[4s]$. □

# 4 Lower Bounds for Explicit Functions

## 4.1 (DeMorgan) Linear Algebra Method: $\mathcal{L}(PARITY_n) \in \Omega(n^2)$

Let us define $PARITY_n$ as $\bigoplus_n$ and $1 - PARITY_n$ as $\overline{\bigoplus}_n$. Recall[4] that $\mathcal{C}(\bigoplus_n) \leq 3(n-1)$ and $\mathcal{L}(\bigoplus_n) \leq 2^{2\lceil \log n \rceil}$. We will show that these bounds are tight.

Notation: $\lambda(\mathbf{P})$ is the largest eigenvalue of a symmetric matrix $\mathbf{P}$. Recall[5] that

$$\lambda(\mathbf{P} + \mathbf{Q}) \leq \lambda(\mathbf{P}) + \lambda(\mathbf{Q}).$$

For non-empty $A, B \subseteq \{0,1\}^n$, the matrix $\mathbf{M} \subseteq \{0,1\}^{A \times B}$ is the matrix

$$\mathbf{M}_{a,b} = \begin{cases} 1 & \text{if } a_i \neq b_i \text{ for exactly one } i \\ 0 & \text{otherwise} \end{cases}$$

you can read this as "the hamming distance of $\mathbf{a}$ and $\mathbf{b}$ differs by exactly one". Note that $\mathbf{M}^\intercal\mathbf{M} \in \mathbb{N}^{B \times B}$ with entry $(i,j)$ interpreted as "the number of vectors $\mathbf{a} \in A$ such that both $\mathbf{b}_i$ and $\mathbf{b}_j$ are one away from $\mathbf{a}$". Similarly $\mathbf{M}\mathbf{M}^\intercal \in \mathbb{N}^{A \times A}$ with entry $(i,j)$ interpreted as "the number of vectors $\mathbf{b} \in B$ such that both $\mathbf{a}_i$ and $\mathbf{a}_j$ are one away from $\mathbf{b}$". It is a fact from linear algebra that $\mathbf{M}^\top\mathbf{M}$ and $\mathbf{M}\mathbf{M}^\top$ have the same non-zero eigen-values. In particular, $\lambda(\mathbf{M}^\intercal\mathbf{M}) = \lambda(\mathbf{M}\mathbf{M}^\intercal)$.

**Theorem 17.** ***(Koutsoupias 1993).*** *For any $f : \{0,1\}^n \to \{0,1\}$, $A \subseteq f^{-1}(0)$, and $B \subseteq f^{-1}(1)$,*

$$\mathcal{L}(f) \geq \lambda(\mathbf{M}^\intercal\mathbf{M}).$$

*Proof.* By induction on $\mathcal{L}(f)$. The base case occurs when $\mathcal{L}(f) = 1$ and the circuit only reads in one out of the $n$ variables of the input. W.l.o.g assume that the input to the leaf is $x_1$. Then $f(\mathbf{x}) = x_1$ or $f(\mathbf{x}) = 1 - x_1$; assume the former. Let $A = f^{-1}(0)$ and $B = f^{-1}(1)$. Then $A = \{0s : s \in \{0,1\}^{n-1}\}$ and $B = \{1s : s \in \{0,1\}^{n-1}\}$. Recall that entry $(i,j)$ of $\mathbf{M}^\top\mathbf{M}$ is the number of elements $\mathbf{a} \in A$ such that both $\mathbf{b}_i$ and $\mathbf{b}_j$ differ from $\mathbf{a}$ by one. Notice that $\mathbf{a} = 0s$ and $\mathbf{b} = 1s'$ differ by exactly one if and only if $s = s'$. Thus $\mathbf{M}^\top\mathbf{M}$ is exactly the identity matrix with dimension $|B| \times |B|$ and $\lambda(\mathbf{M}^\top\mathbf{M}) = 1$ satisfying the theorem.

[4]Construct a circuit with $n-1$ $\oplus$-gates and substituting three DeMorgan gates $(x \wedge \neg y) \vee (\neg x \wedge y)$ for each $x \oplus y$.

[5]I think this can be shown as follows. Take the largest eigen-vector $\mathbf{x}$ of $\mathbf{P}$ and decompose it in the eigen-basis of $\mathbf{Q}$. Then right-multiplying $\mathbf{P} + \mathbf{Q}$ by $\mathbf{x}$.

In the inductive step, let $F$ be a formula which computes $f$ of size $\mathcal{L}(f)$. Suppose that $F = F_1 \wedge F_2$ for some circuits $F_1$ and $F_2$. Let $f_1$ and $f_2$ be the functions computed by $F_1$ and $F_2$ respectively. Notices that $\mathcal{L}(f) = \mathcal{L}(f_1) + \mathcal{L}(f_2)$. Let $A_1 = f_1^{-1}(0)$ and $A_2 = A \backslash A_1$. Since $F = F_1 \wedge F_2$, $A_2 \subset f_2^{-1}(0)$ as at least one of $F_1$ or $F_2$ must evaluate to 0. Consider matrices $\mathbf{M}_1 \in \mathbb{N}^{A_1 \times B}$ and $\mathbf{M}_2 \in \mathbb{N}^{A_2 \times B}$. Notice that $\mathbf{M}^\top \mathbf{M} = \mathbf{M}_1^\top \mathbf{M}_1 + \mathbf{M}_2^\top \mathbf{M}_2$ since $A_1 \cup A_2 = A$ and each matrix product counts the number of off-by-one vectors $\mathbf{a}$ matched to by $\mathbf{b} \in B$. Then

$$\begin{aligned}
\lambda(\mathbf{M}^\top \mathbf{M}) &= \lambda(\mathbf{M}_1^\top \mathbf{M}_1 + \mathbf{M}_2^\top \mathbf{M}_2) && \text{(definition)}\\
&\leq \lambda(\mathbf{M}_1^\top \mathbf{M}_1) + \lambda(\mathbf{M}_2^\top \mathbf{M}_2) && \text{(symmtric matrix prop.)}\\
&\leq \mathcal{L}(f_1) + \mathcal{L}(f_2) && \text{(induction hyp.)}\\
&= \mathcal{L}(f)
\end{aligned}$$

The same is true if $F = F_1 \vee F_2$, but this requires decomposing $B$. Remember however that $\lambda(\mathbf{M}^\top \mathbf{M}) = \lambda(\mathbf{M}\mathbf{M}^\top)$ so it does not make much of a difference. □

**Corollary 18.** ***(Khrapchenko 1971).***

$$\mathcal{L}(f) \geq \frac{\left(\sum_{\mathbf{a} \in A} \sum_{\mathbf{b} \in B} \mathbf{M}_{a,b}\right)^2}{|A| \cdot |B|}$$

*Proof.* [6] The idea is to write $\lambda(\mathbf{M}^\top \mathbf{M})$ as a Rayleigh quotient and then substitute in $\mathbb{1}$ to get that lower bound.

$$\begin{aligned}
\lambda\left(\mathbf{M}^\top \mathbf{M}\right) &= \max_{\mathbf{z} \in \mathbb{R}^B \backslash \emptyset} \frac{\mathbf{z}^\top \mathbf{M}^\top \mathbf{M} \mathbf{z}}{\mathbf{z}^\top \mathbf{z}}\\
&\geq \frac{\mathbb{1}^\top \mathbf{M}^\top \mathbf{M} \mathbb{1}}{|B|}\\
&= \frac{\sum_{a \in A} \left(\sum_{b \in B} \mathbf{M}_{a,b}\right)^2}{|B|}\\
&\geq \frac{\left(\sum_{a \in A} \sum_{b \in B} \mathbf{M}_{a,b}\right)^2}{|A| \cdot |B|}
\end{aligned}$$

where the last inequality follows by Cauchy-Schwartz[7]. □

We use the above Corollary 18 to show that $\mathcal{L}(\bigoplus_n) \geq n^2$. Take $A$ and $B$ to be the set of even and odd strings[8] in $\{0,1\}^n$ respectively. Then, by the above,

$$\mathcal{L}(f) \geq \frac{\left(\sum_{a \in A} \sum_{b \in B} \mathbf{M}_{a,b}\right)^2}{|A| \cdot |B|} = \frac{\left(n 2^{n-1}\right)^2}{2^{n-1} \cdot 2^{n-1}} = n^2.$$

This technique can achieve gaps of at most $n^2$. Exercise: (1)[9] prove lower-bound $\mathcal{L}(MAJ_n) \geq \Omega(n^2)$ and (2) can you devise an upper bound of $\mathcal{L}(MAJ_n) \leq n^{O(1)}$.

[6] :) I like this

[7] The application of Cauchy-Schwartz here is subtle. The key is to multiply top and bottom by $\left(\sum_{a \in A} 1^2\right)$ and combine the two sum of squares.

[8] Here the parity of the string $s$ corresponds to the parity of the sum of ones in $s$.

[9] Hint: Take $A = \{s \in \{0,1\}^n : s \text{ has exactly } \lceil n/2 \rceil - 1 \text{ ones}\}$ and $B = \{t \in \{0,1\}^n : t \text{ has exactly } \lceil n/2 \rceil \text{ ones}\}$.

### 4.2 (General) Gate Elimination: $\mathcal{C}(PARITY_n) \in \Omega(n)$

**Definition 19.** *For $i \in [n]$ and $b \in \{0,1\}$ the **1-bit restriction**, $x_i \leftarrow b$ is the $n$-ary function $f^{(x_i \leftarrow b)}$. The substitution can be done* syntactically *for circuits $C$, namely, $C^{(x_i \leftarrow b)}$. The technique is to substitute $x_i \leftarrow b$ and $\overline{x}_i \leftarrow 1-b$ and performing the relevant simplifications.*

There are a couple of observations to note. (1) If $C$ computes $f$, then $C^{(x_i \leftarrow b)}$ computes $f^{(x_i \leftarrow b)}$. (2) If $x_i$ appears below a gate in $C$ then for both settings of $b \in \{0,1\}$, size $\left(C^{(x_i \leftarrow b)}\right) \leq \text{size}(C) - 1$ i.e. any setting of $b$ will knock out one gate in $C$. (2) There exists a setting of $b$ for each gate, such that size $\left(C^{(x_i \leftarrow b)}\right) \leq \text{size}(C) - 2$ i.e. the setting of $b$ knocks out two gates in $C$.

**Theorem 20.** ***(Schnorr 1979).*** *$\mathcal{C}(PARITY_n) \geq 3(n-1)$.*

*Proof.* By induction. The base case where $n = 1$ is trivial. The crucial observation is as follows. If a literal is below $k$ $\wedge/\vee$ gates (of the same type), then there is a setting of the literal such that you can knock out at least $k$ gates. Just think about the different settings of the literal.

Consider any circuit $C$ which calculates the $PARITY_n$ function. Identify three gates in $C$:

1. A gate whose inputs are two literals. Let these be $x_i$ and $x_j$.
2. Pick a literal of the previous gate, say $x_i$. Find another gate with $x_i$ as an input. Suppose such a gate does not exist. Then, by setting $x_j$ appropriately, we could knock out the gate in step 1 and the output would not depend on $x_i$. This would not calculate the $PARITY_n$ function.
3. The gate above the one in step 2. Such a gate exists if the gate from step 2 is not the output of the circuit. Suppose for a contradiction that it was. Then a setting of $x_i$ would fix the output. This would also not calculate the $PARITY_n$ function.

By setting $x_i$ appropriately, we can kills all three gates above. See Figure 5.

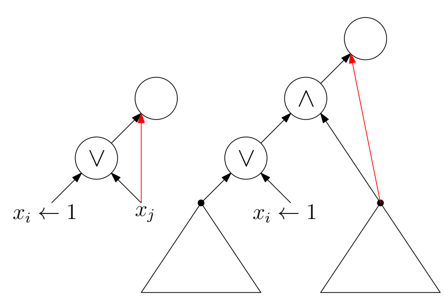


Figure 5: The three gates that get eliminated when we restrict $x_i$. The actual setting of $x_i$ depends on the gate type.

By the induction hypothesis, $C^{(x_i \leftarrow b)}$ has at least $3(n-2)$ gates. Since we were able to eliminate three gates by setting $x_i$, we know that $C$ has to have $3(n-1)$ gates. $\square$

More sophisticated versions of gate elimination allow for slightly better lower bounds. The current record is $5n - o(n)$ for DeMorgan circuits and $\left(3 + \frac{1}{86}\right) n$ for circuits in the full binary basis.

### 4.3 (DeMorgan) Random Restriction: $\mathcal{L}(ANDREEV_{k,m}) \in \Omega(n^3)$

#### 4.3.1 Subbotovskaya's Method

**Definition 21.** *A formula $F$ is* **nice** *if for every sub-formula of the form $x_i \wedge F'$, $\overline{x}_i \wedge F'$, $x_i \vee F'$, $\overline{x}_i \vee F'$, the variable $x_i$ does* not *occur in $F'$.*

**Lemma 22.** *Every formula is equivalent to a* nice *formula of the same (or less) leaf size.*

*Proof.* Given sub-formulas of the form $x_i \wedge F$, $\overline{x}_i \wedge F$, $x_i \vee F$, and $\overline{x}_i \vee F$ where $F$ contains literals $x_i$ or $\overline{x}_i$, repeatedly apply

$$\begin{aligned} x_i \wedge F &\to x_i \wedge F^{(x_i \leftarrow 1)} \\ \overline{x}_i \wedge F &\to \overline{x}_i \wedge F^{(x_i \leftarrow 0)} \\ x_i \vee F &\to x_i \vee F^{(x_i \leftarrow 0)} \\ \overline{x}_i \vee F &\to \overline{x}_i \vee F^{(x_i \leftarrow 1)} \end{aligned}$$

This shows that every minimal formula for a function $f$ is *nice*. □

**Lemma 23.** *For every $f : \{0,1\}^n \to \{0,1\}$,*

$$\mathop{\mathbb{E}}_{i\in[n],b\in\{0,1\}}\left[\mathcal{L}\left(f^{(x_i\leftarrow b)}\right)\right] \leq \left(1-\frac{1}{n}\right)^{1.5}\mathcal{L}(f).$$

*Proof.* Let $F$ be a minimal nice formula for $f$. Let $\ell_i$ be the all leaves of $F$ labelled with $x_i$ or $\bar{x}_i$. Then $\mathcal{L}(f) = \sum_{i=1}^{n} \ell_i$. Notice that every gate $g$ with a leaf $\lambda$ has an associated sub-formula $F'$ such that $\lambda$ does not occur in $F'$.

For a bit $b \in \{0,1\}$, the random restriction $F^{(x_i\leftarrow b)}$ will kill leaf $x_i$ with probability 1 and kill all leaves in $F'$ with probability $\frac{1}{2}$. Thus in expectation, 1.5 leaves are killed under the 1-bit restriction $F^{(x_i\leftarrow b)}$. For each $i \in [n]$ we have

$$\mathop{\mathbb{E}}_{b\in\{0,1\}}\left[\mathcal{L}(F) - \mathcal{L}\left(F^{(x_i\leftarrow b)}\right)\right] \geq 1.5\ell_i.$$

Averaging over all choices of $i$, we have that

$$\mathop{\mathbb{E}}_{i\in[n],b\in\{0,1\}}\left[\mathcal{L}(F) - \mathcal{L}\left(F^{(x_i\leftarrow b)}\right)\right] \geq \frac{1.5}{n}\sum_{i=1}^{n}\ell_i = \frac{1.5\mathcal{L}(F)}{n}.$$

Rearranging the above, we have

$$\mathop{\mathbb{E}}_{i\in[n],b\in\{0,1\}}\left[\mathcal{L}\left(F^{(x_i\leftarrow b)}\right)\right] \leq \left(1-\frac{1.5}{n}\right)\mathcal{L}(F) \leq \left(1-\frac{1}{n}\right)^{1.5}\mathcal{L}(F)$$

where the last inequality follows as $1 - ax \leq (1-x)^a$. □

Apparently, this lemma implies that $\mathcal{L}(\oplus_n) \geq n^{1.5}$.

**Definition 24.** *A* ***restriction*** *$\rho$ is a function $\rho : [n] \to \{0, 1, *\}$ which can be thought of as a partial assignment of an $n$-ary Boolean function $f$. Denote the restriction of $f$ under $\rho$ as $f \upharpoonright \rho : \{0,1\}^{\rho^{-1}(*)} \to \{0,1\}$. Further $\rho$ is a* ***$k$-star restriction*** *if $|\rho^{-1}(*)| = k$.*

*Let $p \in [0, 1]$. In a* ***$p$-random restriction*** *where you set*

$$R_p(i) = \begin{cases} * & \textit{with probability } p \\ 0 & \textit{with probability } \frac{1-p}{2} \\ 1 & \textit{with probability } \frac{1-p}{2} \end{cases}$$

**Theorem 25.** *Let $f : \{0,1\}^n \to \{0,1\}$ and let $\rho$ be a uniform random $k$-start restriction. Then*

$$\mathbb{E}\left[\mathcal{L}(f \upharpoonright \rho)\right] \leq \left(\frac{k}{n}\right)^{1.5} \mathcal{L}(f).$$

*Proof.* Repeatedly apply Lemma 23 to restrict $k$ bits to get

$$\mathbb{E}\left[f \upharpoonright \rho\right] \leq \left(1 - \frac{1}{n}\right)^{1.5} \cdot \left(1 - \frac{1}{n-1}\right)^{1.5} \cdots \left(1 - \frac{1}{k+1}\right)^{1.5} \mathcal{L}(f) = \left(\frac{k}{n}\right)^{1.5} \mathcal{L}(f).$$

$\square$

**Corollary 26.** ***(Subbotovskaya 1961).***

$$\mathbb{E}\left[\mathcal{L}\left(f \upharpoonright R_p\right)\right] \leq O\left(p^{1.5}\mathcal{L}(f) + 1\right).$$

According to Håstad (19 something or other) and Tal (2014), this can be improved to $O(p^2\mathcal{L}(f)+1)$.

**Open problem**: what is the shrinkage exponent of monotone formulas? (this is known to be between 2 and $\left(\log(\sqrt{5}) - 1\right)^{-1} = 3.27$).

### 4.3.2 DEF: Composition of Boolean Functions

**Definition 27.** *Let $f : \{0,1\}^k \to \{0,1\}$ and $g : \{0,1\}^m \to \{0,1\}$. Let $f \otimes g : (\{0,1\}^m)^k \to \{0,1\}$ is defined as*

$$(f \otimes g)(\mathbf{x}_1, ..., \mathbf{x}_k) = f\left(g(\mathbf{x}_1), ..., g(\mathbf{x})\right).$$

*In essence the composition is of the form $f \otimes g = f \circ g^k$.*

Think of the input of the composition as a matrix $\mathbf{X} \in \{0,1\}^{k \times m}$ with rows $\mathbf{x}_1, ..., \mathbf{x}_k$. Apply $g$ to each row, then apply $f$ to the resulting column vector. Observe that $\mathcal{L}(f \otimes g) \leq \mathcal{L}(f) \cdot \mathcal{L}(g)$.

**Conjecture 28.** ***(KRW).*** *For all functions $f$ and $g$,*

$$\mathcal{L}\left(f \otimes g\right) = \tilde{\Omega}\left(\mathcal{L}(f) \cdot \mathcal{L}(g)\right)$$

*where $\tilde{\Omega}(t(n)) = \Omega(t(n)) / \left(\log t(n)\right)^{O(1)}$ for any function $t(n)$.*

The following is an explicit $n$-ary Boolean function for which the lower bound is true.

**Lemma 29.** *For all $k, m \geq 1$ and $f : \{0,1\}^k \to \{0,1\}$,*

$$\mathcal{L}\left(f \otimes XOR_m\right) \geq \mathcal{L}(f) \cdot \Omega\left(\left(\frac{m}{\log k}\right)^2\right).$$

*Proof.* Let $p = \frac{2\ln k}{m}$. Apply $R_p$ on $k \times m$ variables of $f \otimes XOR_m$. If $R_p$ has a $*$ in every row then

$$\mathcal{L}\left((f \otimes XOR_m) \restriction R_p\right) \geq \mathcal{L}(f)$$

since a formula which calculates the LHS can be used to calculate the RHS. In particular, if there is a $*$ in some row $i$, then from the perspective of $XOR_m$, the value of row $i$ is undetermined. If every single row is undetermined, then the input to $f$ is undetermined. Thus $(f \otimes XOR_m) \restriction R_p$ would be able to compute $f(s)$ for any $s \in \{0,1\}^k$.

Let $E$ be the event that there exists a $*$ in every row of the input matrix after applying $R_p$. We bound $\Pr[E]$ below by bounding $\Pr\left[\overline{E}\right]$ above. Let $B_i$ be the event that some row $i$ is *bad* i.e. does not have a $*$ after applying $R_p$. Since every element is fixed with probability $1-p$, $\Pr[B_i] = (1-p)^m$ for all $i \in [k]$. Then

$$\Pr\left[\overline{E}\right] \leq \sum_{i=1}^{k} \Pr[B_i] = k(1-p)^m \approx k\exp(-pm) \leq \frac{1}{k}$$

where the first inequality follows by union-bound and the last by the definition of $p$ above. Thus we have the following lower bound

$$1 - \frac{1}{k} \leq \Pr\left[E\right] \tag{1}$$

To get an upper bound for $\Pr[E]$, observe that $\Pr[E] \leq \Pr\left[\mathcal{L}\left((f \otimes XOR_m) \restriction R_p\right) \geq \mathcal{L}(f)\right]$. Apply Markov's inequality to get

$$\Pr\left[\mathcal{L}\left((f \otimes XOR_m) \restriction R_p\right) \geq \mathcal{L}(f)\right] \leq \frac{\mathbb{E}\left[\mathcal{L}\left((f \otimes XOR_m) \restriction R_p\right)\right]}{\mathcal{L}(f)}$$

By the improvement noted after Corollary 26, $\mathbb{E}\left[f \restriction R_p\right] \leq O\left(p^2\mathcal{L}(f) + 1\right)$, we have

$$\Pr[E] \leq \frac{\mathbb{E}\left[\mathcal{L}\left((f \otimes XOR_m) \restriction R_p\right)\right]}{\mathcal{L}(f)} = O\left(\frac{p^2\mathcal{L}(f \otimes XOR_m) + 1}{\mathcal{L}(f)}\right). \tag{2}$$

Combining the lower bound from Equation (1) and the upper bound from Equation (2), we have

$$1 - \frac{1}{k} \leq \Pr[E] \leq \frac{p^2\mathcal{L}(f \otimes XOR_m) + 1}{\mathcal{L}(f)}$$

Rearrange with respect to $\mathcal{L}(f \otimes XOR_m)$, taking care to observe that $1 - \frac{1}{k} - \frac{1}{\mathcal{L}(f)} \in O(1)$, to obtain

$$\mathcal{L}\left(f \otimes XOR_m\right) \geq \frac{\mathcal{L}(f)}{p^2} = \mathcal{L}(f) \cdot \Omega\left(\left(\frac{m}{\log k}\right)^2\right)$$

as required. $\square$

### 4.3.3 FUN: $ANDREEV_{k,m}$

Let us construct an explicit function with cubic lower bound on the leaf size using Lemma 29.

**Definition 30.** *For parameters $k, m \in \mathbb{N}$,*

$$ANDREEV_{k,m} : \{k\text{-ary Boolean function}\} \times \{0,1\}^{k\times m} \to \{0,1\}$$

*such that $ANDREEV(f, \mathbf{X}) = (f \otimes XOR_m)(\mathbf{X})$.*

*Think of this as follows: consider the $(k+1)\times 2^k$ table $T$ of $k$-ary Boolean strings and the evaluation of $f$ on these strings. See Table 1. The input matrix $\mathbf{X} \in \{0,1\}^{k\times m}$. Apply the $XOR_m$ function to each row of $\mathbf{X}$ to obtain a $k$-bit string $s$. Find the column of $T$ corresponding to $s$ and return $f(s)$.*

| $f(0^k)$ | $\cdots$ | $f(1^k)$ |
|---|---|---|
| 0 | $\cdots$ | 1 |
| $\vdots$ | $\vdots$ | $\vdots$ |
| 0 | $\cdots$ | 1 |

Table 1: Table $T$ of function $f$.

When $m = 1$, $ANDREEV_{k,1}$ is just the multiplexor function. Let $n = 2^k + mk$. Then $ANDREEV_{k,m}$ can be thought of as an $n$-ary Boolean function with $\mathcal{C}(ANDREEV_{k,m}) = O(n)$.

**Theorem 31.** *For every $f : \{0,1\}^k \to \{0,1\}$ we have*

$$\mathcal{L}\left(ANDREEV_{k,m}\right) \geq \mathcal{L}\left(f \otimes XOR_m\right) \geq \mathcal{L}(f) \cdot \Omega\left(\left(\frac{m}{\log k}\right)^2\right).$$

*Proof.* By fixing $2^k$ values $f(s)$ for $s \in \{0,1\}^k$, in the formula for $ANDREEV_{k,m}$, we can calculate $f \otimes XOR_m$. Thus $\mathcal{L}(ANDREEV_{k,m}) \geq \mathcal{L}(f \otimes XOR_m)$. By Shannon's Theorem 15, there exists a $k$-ary Boolean function $f$ with circuit size, and thus leaf size, $\Omega(2^k/k)$. Let $m = 2^k/k$ and note that $n = 2^k + mk \in \Theta(2^k)$. Then, by Lemma 29,

$$\mathcal{L}(ANDREEV_{k,m}) \geq \Omega\left(\frac{2^k}{k}\right) \cdot \Omega\left(\left(\frac{m}{\log k}\right)^2\right) = \Omega\left(\frac{n^3}{(\log n)^3 (\log\log n)^2}\right).$$

Thus $\mathcal{L}(ANDREEV_{k,m}) \in \tilde{\Omega}(n^3)$. $\square$

This lower bound for the $ANDREEV_{m,k}$ is nearly tight since $\mathcal{L}(ANDREEV_{k,m}) \in \tilde{O}(n^3)$.

## 4.4 (Full Binary) Subset Subfunction: $\mathcal{L}(ED_n) \in \Omega(n^2)$

### 4.4.1 DEF: $V$-Subfunctions

Let $B_2$ be the full binary basis (all 2-ary gate types). Unfortunately, the random restriction idea does not work in this setting since it is *not true* that

$$\mathbb{E}[\mathcal{L}_{B_2}(f \upharpoonright R_p)] \leq O\left(p^{1+\epsilon}\mathcal{L}_{B_2}(f) + 1\right)$$

for any $\epsilon > 0$. Do you see why?[10]

**Definition 32.** *For $f : \{0,1\}^n \to \{0,1\}$ and $V \subset [n]$,*

$$\text{sub}_V(f) = \{f \restriction \rho : \rho : [n] \to \{0,1,*\} \text{ such that } \rho^{-1}(*) = V\}$$

*be the set of V-**subfunctions** of f.*

*Further, define*

$$\text{sub}^*_V(f) = \{f', 1 - f', \underline{0}, \underline{1} : f' \in \text{sub}_V(f)\}$$

*where $\underline{b}$ is the constant $b$ function for $b \in \{0,1\}$. Note that $|\text{sub}^*_V(f)| \leq 4 \cdot |\text{sub}_V(f)|$ (actually $|\text{sub}^*_V(f)| \leq 2 \cdot |\text{sub}_V(f)| + 2$).*

*Let $F$ be an n-ary formula and $V \subset [n]$ as before. Then **the number of leaves of $F$ labelled by variables in** $V$ be denoted $\ell_V(F)$. Note that $\mathcal{L}(F) = \ell_V(F) + \ell_{[n]\setminus V}(F)$.*

**Example 33.** *Consider $MAJ_3(x_1, x_2, x_3)$ and $V = \{1,2\}$. Then*

$$\text{sub}_V(MAJ_3) = \{x_1 \wedge x_2, x_1 \vee x_2\}$$

*when $x_3$ is restricted to 0 and 1 respectively.*

### 4.4.2 Nechiporuk's Bound

Here are two important properties to note:

1. Suppose $F = \mathsf{gate}(G, H)$ for some $\mathsf{gate} : \{0,1\}^2 \to \{0,1\}$. Then

$$\text{sub}_V(F) \subseteq \{\mathsf{gate}(g, h) : g \in \text{sub}_V(G) \text{ and } h \in \text{sub}_V(H)\}.$$

   and $|\text{sub}^*_V(F)| \leq |\text{sub}^*_V(G)| \cdot |\text{sub}^*_V(H)|$. This should be pretty obvious. Let $f_V$ be any function in $\text{sub}_V(f)$. Then this function must be equivalent to the composition of the function computed by $\mathsf{gate}$ and some two functions $g \in \text{sub}_V(G)$ and $h \in \text{sub}_V(H)$.

2. Suppose that $F = \mathsf{gate}(G, H)$ and $\ell_V(H) = 0$. Then $\text{sub}^*_V(F) \subseteq \text{sub}^*_V(G)$. Note that $\ell_V(H)$ means that none of the leaves in $H$ are labelled with any indices from $V$. When considering the $V$-functions of $H$, these can only be the constant functions $\underline{0}$ and $\underline{1}$. When composing the function calculated by $\mathsf{gate}$ with some $g \in \text{sub}_V(G)$ and a function in $\{\underline{0}, \underline{1}\}$ we can only get $\{g, 1 - g, \underline{0}, \underline{1}\}$. Thus every function in $\text{sub}^*_V(F)$ is also in $\text{sub}^*_V(G)$.

**Lemma 34.** *If $F$ is an n-ary formula, $V \subseteq [n]$, and $\ell_V(F) \geq 1$, then*

$$|\text{sub}^*_V(F)| \leq 4 \cdot 16^{\ell_V(F)-1}.$$

*Proof.* By induction on $\mathcal{L}(F)$. The base case where $\mathcal{L}(F) = 1$ is trivial. The inductive case is also not that bad considering the two observations above. Suppose $F = \mathsf{gate}(G, H)$. If one of

[10] Let $f$ be the parity function.

$\ell_V(G) = 0$ or $\ell_V(H) = 0$, then we can use the second observation. W.l.o.g assume $\ell_V(H) = 0$. By the induction hypothesis we have that

$$|\mathrm{sub}^*_V(F)| \le |\mathrm{sub}^*_V(G)| \le 4 \cdot 16^{\ell_V(G)-1} \le 4 \cdot 16^{\ell_V(F)-1}.$$

Next suppose that $\ell_V(G) \ge 1$ and $\ell_V(H) \ge 1$. Then by the induction hypothesis, we have that

$$|\mathrm{sub}^*_V(F)| \le |\mathrm{sub}^*_V(G)| \cdot |\mathrm{sub}^*_V(G)| = 16^{\ell_V(G)+\ell_V(H)-1} = 16^{\ell_V(F)-1} \le 4 \cdot 16^{\ell_V(F)-1}$$

as required. □

**Corollary 35.** *Let $F$ and $V$ be as above, then $|\mathrm{sub}_V(F)| \le 16^{\ell_V(F)}$.*

This is immediate when $\mathcal{L}(F) \ge 1$. Only $\underline{b}$, $b \in \{0,1\}$, have leaf-size 0, but $|\mathrm{sub}_V(\underline{b})| = 1 \le 16^0$.

**Theorem 36.** ***(Nechiporuk's Bound).*** *For any $f : \{0,1\}^n \to \{0,1\}$ and any partition of $[n]$ into $t$ disjoint components $V_1 \uplus \cdots \uplus V_t$*

$$\mathcal{L}_{B_2}(f) \ge \frac{1}{4}\sum_{i=1}^{t} \log|\mathrm{sub}_{V_i}(f)|.$$

*Proof.* Direct application of Corollary 35. Let $F$ be the minimal formula computing $f$. Then

$$\mathcal{L}_{B_2}(f) = \sum_{i=1}^{t} \ell_{V_i}(F) \ge \sum_{i=1}^{t} \log_{16}|\mathrm{sub}_{V_i}(F)| = \frac{1}{4}\sum_{i=1}^{t} \log|\mathrm{sub}_{V_i}(F)|.$$

□

### 4.4.3 FUN: Element Distinctness $ED_n$

Let us apply Theorem 36 to an explicit function in the full binary basis to get a lower bound.

**Definition 37.** *For $k \in \mathbb{N}$, let $n = 2^k \cdot 2k$. The **element distinctness** function $ED_n$ is*

$$ED_n : \{0,1\}^{2^k \times 2k} \to \{0,1\}$$

*where*

$$ED_n(X_1, ..., X_{2^k}) = \begin{cases} 1 & \text{if } X_1, ..., X_{2k} \text{ are distinct elements of } \{0,1\}^{2k} \\ 0 & \text{otherwise} \end{cases}$$

*Think of this as being given $2^k$ binary strings of length $2k$ and asked if they are all distinct.*

**Theorem 38.**

$$\mathcal{L}_{B_2}(ED_n) = \Omega\left(\frac{n^2}{\log n}\right).$$

*Proof.* Apply Nechiporuk's bound with $V_i$ as a block of length $2k$ corresponding to the coordinates of $X_i$. Remember $n = 2^k + 2k$. □

Exercise: show that $\Omega(n^2/\log n)$ is the limit on the lower bound achievable by Nechiporuk's method.

## 5 Non-uniformity is More Powerful than Randomness

**Definition 39.** *A* ***randomized circuit*** *for a function* $f : \{0,1\}^n \to \{0,1\}$ *is a circuit* $C$ *with* $n+m$ *variables* $x_1, ..., x_n$ *and* $y_1, ..., y_m$ *(think of* $\mathbf{x}$ *as the input and* $\mathbf{y}$ *as a random seed) such that for every* $\mathbf{x} \in \{0,1\}^n$

$$\Pr_{\mathbf{y}\in\{0,1\}^m}[C(\mathbf{x},\mathbf{y})=1] \begin{cases} \geq \frac{2}{3} & \text{if } f(\mathbf{x})=1 \\ \leq \frac{1}{3} & \text{if } f(\mathbf{x})=0 \end{cases}$$

Let $\mathsf{BPP/poly}$ be the class of Boolean functions computable by poly-sized randomized circuits — think of $\mathsf{BPP/poly}$ as the non-uniform version of $\mathsf{BPP}$. Generally $\frac{1}{3}$ and $\frac{2}{3}$ can be replaced with any $a$, $b$ satisfying $0 < a < b < 1$.

**Theorem 40.** ***(Adelman 1978).*** *If* $f$ *is computable by poly-size randomized circuit, then it is computable by poly-sized (deterministic) circuits i.e.* $\mathsf{BPP/poly} \subseteq \mathsf{P/poly}$.

*Proof.* The intuition is to improve the probability of success by doing repeated trials, taking the majority, then use the probabilistic method to show that there was a good choice of $\mathbf{y}$ which we can hard-wired into the circuit.

Let $f$ be the given $n$-ary function with $\mathsf{BPP/poly}$ circuit $C(\mathbf{x},\mathbf{y})$. Recall that $MAJ_k$ has $O(k)$ sized circuits. Construct the composite function $g_k : \{0,1\}^n \times \{0,1\}^{k\times m} \to \{0,1\}$ such that

$$g_k(\mathbf{x},\mathbf{Y}) = MAJ_k\left(C(\mathbf{x},\mathbf{y}_1), ..., C(\mathbf{x},\mathbf{y}_k)\right)$$

where each $\mathbf{y}_i$ is the $i^{\text{th}}$ row of $\mathbf{Y}$. Observe that $\mathcal{C}(g_k) \leq k \cdot \mathcal{C}(f) + O(k)$ since we can replace each of the $k$ inputs of $MAJ_k$ by a circuit of size $\mathcal{C}(f)$. If $k$ is poly$(n)$ then $g_k \in \mathsf{P/poly}$.

On a randomly sampled seed $\mathbf{y}_i$, let $X_i$ be the indicator r.v. for $C(\mathbf{x},\mathbf{y}_i) \neq f(\mathbf{x})$. Let $X = X_1 + \cdots + X_k$. Observe that

$$\begin{aligned}\Pr_{\mathbf{Y}\in\{0,1\}^{k\times m}}[g_k(\mathbf{x},\mathbf{Y}) \neq f(\mathbf{x})] &= \Pr_{\mathbf{y}_1,...,\mathbf{y}_k\in\{0,1\}^m}\left[MAJ_k\left(C(\mathbf{x},\mathbf{y}_1),...,C(\mathbf{x},\mathbf{y}_k)\right) \neq f(\mathbf{x})\right] \\ &= \Pr\left[X \geq \frac{k}{2}\right] \\ &= \Pr\left[X \geq (1+\epsilon)pk\right]\end{aligned}$$

where $p = \Pr[C(\mathbf{x},\mathbf{y}) \neq f(\mathbf{x})]$ and $\epsilon = \frac{1-2p}{2p}$. From Definition 39, we have $p = \frac{1}{3}$ and $\epsilon = \frac{1}{2}$. Thus by Chernoff bound we have

$$\Pr_{\mathbf{Y}\in\{0,1\}^{k\times m}}[g_k(\mathbf{x},\mathbf{Y}) \neq f(\mathbf{x})] = \Pr\left[X \geq (1+\epsilon)pk\right] \leq \exp\left(\frac{-\epsilon^2 pk}{2+\epsilon}\right) = \exp\left(\frac{-k}{30}\right).$$

When $k > 30$, $\Pr[g_k(\mathbf{x},\mathbf{Y}) \neq f(\mathbf{x})] \leq 2^{-n}$ and there exists a $\mathbf{Y}$ for which $g_k(\mathbf{x},\mathbf{Y}) = f(\mathbf{x})$ with probability greater than $1-2^{-n}$. Since there are only $2^n$ inputs $\mathbf{x}$, $g_k(\mathbf{x},\mathbf{Y})$ matches $f(\mathbf{x})$ on every input. Hard-wiring $\mathbf{Y}$ into the circuit for $g_k$ produces a deterministic poly$(n)$ circuit for $f$. ☐

## 6 Restricted Setting

Welp, that is all the bounds that we could get out of the general case. Time to consider some restricted settings.

## 6.1 Monotone Circuits

**Definition 41.** *A* ***monotone circuit***

### 6.1.1 Upper Bound: Majority

Observe that if $MAJ_n(\mathbf{x}) = 0$, then for some uniformly random bit $x_i$, $\Pr[x_i = 1] = \frac{1}{2} - \frac{1}{2n}$.

**Theorem 42.** ***(Valiant 1984).*** *$MAJ_n$ has poly-sized monotone circuits*[11]*.*

*Proof.* We are going to use the idea of amplification and projections. The idea is to compose $MAJ_3$ with itself $k$ times. ☐

**Definition 43.** *For $f : \{0,1\}^m \to \{0,1\}$, let $\mu_f : [0,1] \to [0,1]$ be defined as*

$$\mu_f(p) = \Pr_{y_1,...,y_m \in Bern(p)} \Pr[f(y_1, ..., y_m) = 1].$$

$\mu$ is particularly nice for monotone, non-constant functions $f$.

**Example 44.** $MAJ_3(p) = p^3 + 3p^2(1 - p)$.

Observe that $\mu_{f \otimes g}(p) = \mu_f(\mu_g(p))$. To see this, you should draw out the little tree.

**Lemma 45.** *There is a constant $c < 3$ such that $\mu_{MAJ_3}^{c \log n}\left(\frac{1}{2} - \frac{1}{2n}\right)$*

A striking consequence of this result.

**Definition 46.** *$f : \{0,1\}^n \to \{0,1\}$ is a* ***Slice functions*** *if there exists $k \in \{0, ..., n\}$ such that $f(x) = 0$ if $|x| < k$ and $f(x) = 1$ if $|x| > k$.*

**Theorem 47.** ***(Berkowitz 1982).*** *If $f$ is a slice function then...*

## 6.2 Bounded Depth Circuits: $\mathsf{AC}^0$

**Definition 48.** $\mathsf{AC}^0$

**Challenge.** Let $\mathcal{L}_d(f)$ be the leaf size of an $n$-ary Boolean function $f$ in $\mathsf{AC}^0$ with depth at most $d$. Observe that $\mathcal{L}_2(PARITY_n) \leq n2^n$ as you can take an "or" of $2^n$ "and" gates with fan-in $n$ each specifying a setting of the $n$ input variables.

Further note that for $k$ and $n_1, ..., n_k$ where $\sum_{i=1}^{k} n_i = n$,

$$\mathcal{L}_{d+1}(PARITY_n) \leq 2^{k-1} \sum_{i=1}^{k} \mathcal{L}_d(XOR_{n_i}). \tag{3}$$

[11] And monotone formulas apparently!

(I still need to work out the details of this proof). Using $\mathcal{L}_2(PARITY_n) \leq n2^n$ for the base case and the recurrence shown in Equation (3), we can show

$$\mathcal{L}_{d+1}(PARITY_n) \leq n2^{dn^{1/d}}$$

for any $n$ and $d \geq 2$. However, when $n$ is a power of 2, we can get a slightly tighter bound of

$$\mathcal{L}_{d+1}(PARITY_n) \leq n2^{d\left(n^{1/d}-1\right)}.$$

Ben suspects that the above inequality holds for all $n$, not just powers of two, since for $d = \lceil \log n \rceil$ it is know that $n^{1/d} - 1 = 2^{\log n / \log n} - 1 = 1$ in the exponent of 2 is sufficient i.e. it is known that

$$\mathcal{L}_{\lceil \log n \rceil}(PARITY_n) \in O(n^2).$$

Further, Ben believes that it is sufficient to achieve this tighter bound by analyzing the recurrence relation more carefully.

## 7 Håstad's Switching Lemma

**Definition 49.** *A **decision tree** (DT) is a rooted binary tree whose leaves are labelled by $\{0,1\}$ and whose internal nodes are labelled by variables. The **depth** of a decision tree is the length of the longest root-to-leaf path. For $f : \{0,1\}^n \to \{0,1\}$, let $\mathcal{DT}_{\text{depth}}(f)$ denote depth of the minimum depth DT that computes $f$.*

*It is useful to consider formulas as a DNF or CNF. The **width** of a DNF(CNF) formula is the maximum number of literals in any clause.*

**Definition 50.** *Let $F = C_1 \vee \cdots \vee C_m$ be a k-DNF with an arbitrary fixed order on the clauses and variables. Let $\rho : \{x_1, ..., x_n\} \to \{*, 0, 1\}$ be a restriction. Then the **canonical decision tree** of $F \restriction \rho$, denoted $\mathcal{CDT}(F, \rho)$, is constructed as follows.*

1. *Let $F_1 = F \restriction \rho$ and $C_{1,1} \vee \cdots \vee C_{1,k_1}$ be the set of clauses which have not been set to 1 or 0 by $\rho$.*
2. *Construct a perfect binary tree where each level of the tree is labelled by a in $C_{1,1}$ in the predetermined order e.g. if $C_{1,1} = x_i x_j$ in $F_1$ and $i < j$, then set the root to $x_i$ and label its two children $x_j$.*
3. *Let $p$ be a path in our partially constructed tree. Denote by $\rho_p$ a which sets each variable in $C_{1,1}$ according to the path $p$. Let $\ell$ be one of the $2^{|C_{1,1}|}$ leafs in our partially constructed tree. If $p_\ell$ is the root-to-$\ell$ path, append $\mathcal{CDT}(F', \rho_p)$ to $\ell$.*

*See the following example.*

**Example 51.** ***(Constructing a canonical decision tree).*** *Let our k-DNF be*

$$F = \bar{x}_1 x_3 x_5 \vee x_1 x_2 \bar{x}_3 \vee x_2 \bar{x}_4 x_5 \vee x_3 x_4 \bar{x}_6 \vee x_1 \bar{x}_4 \bar{x}_7$$

*and our restriction $\rho = \{x_1 \mapsto 1, x_4 \mapsto 0\}$. Apply the restriction $\rho$ to $F$, to obtain the following*

$$F \upharpoonright \rho = {\color{red}0} \vee x_2\bar{x}_3 \vee x_2x_5 \vee {\color{red}0} \vee \bar{x}_4\bar{x}_7.$$

*Look at the first un-falsified clause $x_2\bar{x}_3$ and build a tree with these variables on the first and second levels. For every root-to-leaf path, construct a restriction by setting the variables $x_2$ and $\bar{x}_3$ according to the edge labels. Continue down the tree until all the variables have been added. The first two restrictions are shown in Figure 6.*

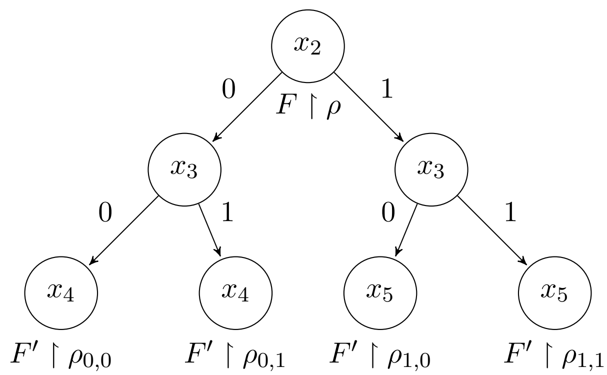


Figure 6: Given the 3-DNF $F$ and the restriction $\rho$ construct the canonical DT for $F \upharpoonright \rho$. Let $F' = F \upharpoonright \rho$ and $\rho_{a,b} = \{x_2 \mapsto a, x_3 \mapsto b\}$.

Observe that every depth $d$ decision tree (DT) is equivalent to a $d$-DNF ($d$-CNF) by tracing every root-to-leaf path in the tree which end in a one (resp. end in a zero then apply DeMorgan's rule). It follows that an $\vee$ of depth $d$ *DT*s is a $d$-DNF. Further, if $f$ is equivalent to both a $k$-DNF and $l$-CNF, then $\mathcal{DT}_{\text{depth}}(f) \le k \cdot l$. Do you see why this is?

Let $f$ be equivalent to the $k$-DNF $F = A_1 \vee \cdots \vee A_s$ and $\bar{f}$ be equivalent to the $l$-DNF $\bar{F} = B_1 \vee \cdots \vee B_t$. Notice that every pair $(A_i, B_j)$ must share a common variable of opposite sign or else there will be a setting of the variable which simultaneously satisfies $F$ and $\bar{F}$. The proof is by induction on the number of variables. When there is only one variable the claim is true. Suppose $f$ has $n$ variables. Build a decision tree of width $A_1$ which considers all variables in the clause $A_1$. Consider a restriction $\rho$ corresponding to a root-to-leaf path in the DT we have just created. The DNF-width of $F \upharpoonright \rho$ is at most $k$. The DNF-width of $\bar{F}$ is *less than* $l$ since at least one variable of each clause was knocked out. Further, by the induction hypothesis, we have

$$\mathcal{DT}_{\text{depth}}(f \upharpoonright \rho) \le \text{DNF}_{\text{width}}(F \upharpoonright \rho) \cdot \text{DNF}_{\text{width}}(\bar{F} \upharpoonright \rho).$$

By appending the restricted decision tree to every leaf of the depth $k$ DT that we created for the variables in $A_1$, we have that

$$\mathcal{DT}_{\text{depth}}(f) \le k + \text{DNF}_{\text{width}}(F \upharpoonright \rho) \cdot \text{DNF}_{\text{width}}(\bar{F} \upharpoonright \rho) \le k + k(l-1) = kl.$$

Before looking at the proof of the Switching Lemma, let us consider a simple warm-up pertaining to the depth of a decision tree hit by a random restriction.

**Theorem 52.** *If $\mathcal{DT}_{\text{depth}}(f) = k$, then*

$$\Pr\left[\mathcal{DT}_{\text{depth}}(f \upharpoonright R_p) \geq t\right] \leq (2p)^t \binom{k}{t}.$$

*Proof.* By induction on $k$. In the base case where $k = 0$, $f$ is a constant function. Suppose that $\mathcal{DT}_{\text{depth}}(f) = d$. Let $T$ be a DT of $f$ of depth $d$ with root labelled $x_1$. Further let $T_0$ and $T_1$ be the left and right sub-trees of $T$ respectively. Either $x_1$ gets restricted or not. If $x_1$ gets restricted, then we can apply the induction hypothesis to one of $T_0$ or $T_1$, w.l.o.g suppose $T_0$. This happens with probability $1-p$. If $x_1$ is unrestricted then it suffices to find the probability that a restriction of $T_0$ or $T_1$ has depth $\geq t-1$; w.l.o.g. suppose $T_0$ is larger. This happens with probability $p$. Thus

$$\begin{aligned}
\Pr\left[\mathcal{DT}_{\text{depth}}(T \upharpoonright R_p) \geq t\right] &= (1-p)\Pr\left[\mathcal{DT}_{\text{depth}}(T_0 \upharpoonright R_p) \geq t\right] + \\
&\qquad p\Pr\left[(\mathcal{DT}_{\text{depth}}(T_0 \upharpoonright R_p) \geq t-1) \vee (\mathcal{DT}_{\text{depth}}(T_1 \upharpoonright R_p) \geq t-1)\right] \\
&\leq (1-p)\Pr\left[\mathcal{DT}_{\text{depth}}(T_0 \upharpoonright R_p)\right] + 2p\Pr\left[\mathcal{DT}_{\text{depth}}(T_0 \upharpoonright R_p) \geq t-1\right] \\
&\leq (1-p)\cdot(2p)^t\binom{d-1}{t} + 2p\cdot(2p)^{t-1}\binom{d-1}{t-1} \\
&\leq (2p)^t\left(\binom{d-1}{t} + \binom{d-1}{t-1}\right) \\
&= (2p)^t\binom{d}{t}
\end{aligned}$$

where the second inequality follows from a union bound. □

**Theorem 53.** ***(Håstad's Switching Lemma).*** *If $F$ is a k-DNF (or k-CNF) then*

$$\Pr[\mathcal{DT}_{\text{depth}}(F \upharpoonright R_p) \geq t] \leq (5pk)^t.$$

*Proof.* This proof due to Razborov as explained in the exposition by Beame. Let[12]

$$B_t := \{\rho : \mathcal{DT}_{\text{depth}}(\mathcal{CDT}(F, \rho)) = t\}.$$

We will show that the probability a random restriction $R_p \in B_t$ is bounded by $O(pk)^t$. Let $n$ be the number of variables and $m$ the number of clauses in $F$.

For each $\rho$ in $BAD_t$ we want to construct a pair $(\rho^*, \textsc{code})$. $\rho^*$ is a fixed restriction setting a further $t$ variables. $\textsc{code}$ is some information needed to uniquely identify $\rho$. Note that $\textsc{code} = (\mathbf{s}, \pi, \mathbf{n})$ where $\mathbf{s} \in [k]^t$ indicates the location of variables within a clause, $\pi \in \{0,1\}^t$ is the appropriate setting of the variable, and $\mathbf{n} \in \{0,1\}^t$ indicates if we move onto the **n**ext clause. Then, with the pair $(\rho^*, \textsc{code})$ and $F$ in-hand, we can reconstruct $\rho$.

In the encoding step we are given the $k$-DNF $F$ and the restriction $\rho \in B_t$. We will build $\rho^* = \rho\sigma_1 \cdots \sigma_j$ and $\textsc{code} = \gamma_1 \cdots \gamma_j$ incrementally.

1. Let $T = T_1 = \mathcal{CDT}(F, \rho)$ and let $C_1, ..., C_{j_1}$ be the set of clauses with free variables. Let $p$ be the lexicographically first path of length $t$ in $T$. Such a longest path exists since $\rho \in B_t$ and $f$ is not the constant function.

[12] In many proofs the set $B_t$ is actually called $BAD_t$. This is illustrative of how you want to think about $B_t$. It is the set of all *bad* restrictions which cannot be converted to a depth $t$ decision tree.

2. Let $\sigma_1$ be the unique setting of the free variables in $C_1$ such that $C_1 \restriction \sigma_1 \equiv 1$. Let $\pi_1$ record the actual setting of the free variables in $C_1$ along path $p$ as well as their location and number. In particular, $\gamma_1 = (\mathbf{s}_1, \pi_1, \mathbf{n}_1)$ which are the indices, settings, and *is-last-in-clause* properties of the free variables in $C_1$.

   Consider this first encoding step on Example 51. Suppose the long path $p$ aims to set $x_2 = 1$ and $x_3 = 1$. Then $\sigma_1 = \{x_2 \mapsto 1, x_3 \mapsto 0\}$, $\mathbf{s}_1 = [2, 3]$, $\pi_1 = \{x_2 \mapsto 1, x_3 \mapsto 1\}$, and $\mathbf{n} = [0, 1]$.

3. Proceed down $p$ past all the free variables in $C_1$. Let $T_1$ be the remaining subtree. Repeat the process until all $t$ variable on $p$ have been considered.

In the decoding step we are given $F$ and the pair $(\rho^*, \text{CODE})$ where $\rho^* = \rho\sigma_1 \cdots \sigma_j$ and $\text{CODE} = \gamma_1 \cdots \gamma_j$ and want to reconstruct $\rho$.

1. Evaluate $F \restriction \rho^*$. Let $C_1'$ be the first clause set equal to 1. Observe that $C_1' = C_1$ by construction. Use $\mathbf{n}_1$ to determine how many variables were set during the encoding step and use $\pi_1$ and $\mathbf{v}_1$ to reset these values in $\rho^*$ so they match the first $|C_1|$ variables on the longest path $p$. Let $\rho_1^* = \rho\pi_1\sigma_2 \cdots \sigma_j$.

2. Repeat with $\rho_1^*$ in the place of $\rho^*$ until we have used up all $t$ variables.

All variables consider in the above procedure were originally stars in $\rho$.

Suppose that $\rho$ has $s$ stars. Then the set $\mathcal{R}_s$ of all restrictions with $s$ stars is of size $\binom{n}{s}2^{n-s}$. Similarly, the set of all restrictions with $s - t$ stars is of size $\binom{n}{s-t}2^{n-s+t}$. Then, since the codes come from a domain of size $(4 \log k)^t$,

$$\Pr[\mathcal{DT}_{\text{depth}}(F \restriction R_p) \geq t] = \frac{|B_t|}{|\mathcal{R}_s|} \leq \frac{|\mathcal{R}_{s-t}|(4 \log k)^t}{|\mathcal{R}_s|} = \frac{\binom{n}{s-t}2^{n-s+t}(4 \log k)^t}{\binom{n}{s}2^{n-s}} \leq (8pk)^t.$$

Considering all bad restrictions with paths of length at least $t$ increases the above probability by a constant. The eight in the upper bound can be improved to five with a more careful analysis. □

It is often natural to choose $p = \frac{1}{10k}$ so that the bound is inverse exponential in $t$.

**Corollary 54.** *If $F$ is $k$-DNF, then*

$$\Pr\left[F \restriction R_p \textit{ is not a } t\textit{-CNF}\right] \leq (5pk)^t.$$

This corollary follows from Theorem 53 since every depth-$t$ DT is a $t$-CNF ($t$-DNF).

## 7.1 (LB) Parity Circuit-size

**Theorem 55.** *Let $C$ be an* $\mathsf{AC}^0$ *circuit of depth $d + 1$ and size $S$. Let $p = 10^{-d-1}(2 \log S)^{-d}$.*

$$\Pr[\mathcal{DT}_{\text{depth}}(C \restriction R_p) \geq \ell] \leq \frac{1}{2^\ell} + \frac{1}{S}.$$

*Proof.* The key idea[13] is to repeatedly apply Theorem 53 being careful to choose appropriate values of $k$, $t$ and $p$. Consider the bottom level of $C$ (closest to the literals) and w.l.o.g assume that this is a conjunction ($\wedge$-gate). Since the fan-in consists of literals, you can think of them as width one clauses. Apply the theorem with $k = 1$, $t = 2\log S$, $p_1 = \frac{1}{10}$. According to the switching lemma, we fail[14] at each $\wedge$-node with probability at most $2^{-2\log S} = S^{-2}$. Taking a union bound over at most $S$ $\wedge$-gates, we fail at this level with probability at most $S^{-1}$. Let $E_1$ be the event that we successfully completed the switch from 1-CNF to $2\log S$-DNF. Suppose $E_1$ occurs. Then we can collapse this layer of $\vee$-gates with the $\vee$-gates of the level above.

Apply the switching lemma a further $d-1$ iterations. At step $i$ set $k = 2\log S$, $t = 2\log S$, and $p_i = \frac{p_{i-1}}{20\log S}$ each time conditioning on the success of the previous iterations. Observe that at the probability of failure at any iteration is at most $S^{-1}$.

Consider our final iteration. We assumed that $d$ iterations have succeeded so we have w.l.o.g. a $\vee$-gate sitting on top of depth $2\log S$ decisions (i.e. width $2\log S$-CNF clauses). For the final application of the switching lemma we will set $k = 2\log S$, $t = \ell$, and $p_{d+1} = \frac{p_d}{20\log S}$. Thus this iteration fail with probability at most $2^{-\ell}$. □

**Corollary 56.** $\mathcal{C}_{d+1}(XOR_n) = 2^{\Omega(n^{1/d})}$.

By Theorem 55, the probability that the restriction has depth $\geq 1$ is $< 1$. Thus there is some restriction which sets $C$ to a constant. Since $p = 10^{-d-1}(2\log S)^{-d}$ and $S = 2^{\Omega(n^{1/d})}$, there is a good likely-hood that some variables are un-restricted.

### 7.2 Switching Lemma for Formulas

The lower bound[15] of $\Omega(dn^{1/d})$ matches the existing upper bound.

## 8 Bounded Depth: $\mathsf{AC}^0[p]$

Goal: want an upper bound on the approximate degree of $AND$, $OR$, and $MOD_p$ functions.

**Definition 57.** *Let* $\mathbf{x} = (x_1, ..., x_n)$ *then* $MOD_p(\mathbf{x}) = 1$ *if and only if* $\sum_{i=1}^n x_i \equiv 0 \mod p$.

*Let* $A \in \mathbb{F}_p[x_1, ..., x_n]$ *be a* ***random polynomial*** *over* $\mathbb{F}_p$*. The* ***degree*** *of* $A$ *is the maximum degree of a polynomial in the support for* $A$ *(think of* $A$ *as a random variable in a distribution over some polynomials).*

*Let* $f : \{0,1\}^n \to \{0,1\}$*.* $A$ *is an* $\epsilon$*-approximating for* $f$ *if*

$$\Pr_A[A(x) \neq f(x)] \leq \epsilon$$

*for every* $x \in \{0,1\}^n$*.*

[13] Actually I am still missing something here... but I would be hard-pressed to say what it is...

[14] That is to say: this node cannot be converted to a short DT under $R_p$.

[15] A result of Ben's!

**Lemma 58.** *If $A$ is an $\epsilon$-approximating random polynomial for $f$, then $\exists \alpha \in \text{supp}(A)$ such that*

$$\Pr_{\mathbf{x}\in\{0,1\}^n}[\alpha(x) \neq f(x)] \leq \epsilon$$

*Proof.* Markov inequality then probabilistic method. □

**Definition 59.** *The **$\epsilon$-approximate degree of** $f$, denoted $\deg_\epsilon(f)$, is the minimum degree of an $\epsilon$-approximating random polynomial of $f$.*

*Note that this value is invariant under negation of the inputs and outputs (please understand why).*

**Lemma 60.** *Suppose $f(x) = g(h_1(x), ..., h_m(x))$. Then for any $\delta, \epsilon_1, ..., \epsilon_m$,*

$$\deg_{\delta+\epsilon_1+\cdots+\epsilon_m}(f) \leq \deg_\delta(g) \max_i \deg_{\epsilon_i}(h_i)$$

*Proof.* Just think through this carefully. I think you can get it. □

Assume that these are all $n$-ary Boolean functions.

$MOD_p$:

**Lemma 61.** *For any $\epsilon > 0$ and $n$,*

$$\deg_\epsilon(MOD_p) \leq \deg_0(MOD_p) \leq p - 1$$

*Proof.* Give me a degree $p-1$ polynomial which exactly computes $MOD_p$ (for an $n$ bit input vector). *Hint: Fermat's Little Theorem.* □

$OR$: Now we need some random polynomials

**Lemma 62.** $\deg_\epsilon(OR) \leq p\left(\log_p\left(\frac{1}{\epsilon}\right) + 1\right)$. *One should note that this upper bound is independent of $n$.*

*Proof.* So you want to calculate the probability that $OR(x_1, ..., x_n) \neq (\lambda_1 x_1 + \cdots + \lambda_n x_n)$ when $\mathbf{x} = \mathbf{0}$ and when $\mathbf{x} \neq \mathbf{0}$ (*Hint: 0 for the former case and $\frac{1}{p}$ in the latter.*). This is not good enough so just boost □

$AND$: This is identical to the argument for the $OR$ gate.

# 9 Project Idea

Shrinkage exponent of monotone formulas.