Starting in this class, we will give another proof that Parity is not in $\mathbf{AC^0}$. In fact, we will be able to show the essentially tight result, that depth-$k$ (unbounded fan-in) circuits for Parity require size $2^{\Omega(n^{1/(k-1)})}$. (This is tight up to the $\Omega(\cdot)$.)

Given that we already saw a proof of this with the slightly weaker $1/4k$ in the exponent, why bother? There are a few reasons. The first is that this result gives us something strong for every $k$, even $k = 3$. The second one is basically “because we can”. It’s so rare to get a strong lower bound at all in complexity theory that it’s worth really exploring the ones you get. The third reason is that the proof will give us a much greater understanding of $\mathbf{AC^0}$ circuits than Razborov-Smolensky does. And this is great, because constant-depth AND-OR circuits with unbounded fan-in are about the strongest class which we can “understand”, or really “get a handle on”. I will remind you that as far as we know, every language in $\mathbf{NEXP}$ — indeed, every language in $\mathbf{EXP^{NP}}$ — can be solved by linear-size constant-depth AND-OR circuits with Mod-6 gates.

1 The Switching Lemma

This new proof that Parity is not in $\mathbf{AC^0}$ was given by Håstad in 1986. It is based on a theorem called Håstad’s Switching Lemma. It is a pretty hard theorem, so it’s ironic that it’s known merely as a “lemma”. To state it, we need to first recall some basic definitions.

1.1 Basic definitions

Definition 1.1. A DNF is an OR (disjunction) of terms, where each term is an AND (conjunction) of literals. E.g.,

$$f = x_1x_3\overline{x}_6 \lor x_2\overline{x}_7 \lor x_5x_6x_8x_{11} \lor \cdots$$

(Here the $\wedge$ signs in each term are omitted.) A DNF is a syntactic object, but we also think of it as computing a function $f : \{0,1\}^n \to \{0,1\}$ in the obvious way. The size of $f$ is the number of terms, and the width of $f$ is the maximum number of literals in a term.

Definition 1.2. Similarly, a CNF is an AND (conjunction) of clauses, each of which is an OR (disjunction) of literals. Its size and width are defined in the same way.

Definition 1.3. A decision tree (DT) is an object that looks like this: \[\text{[draw picture]}\]. It is again a syntactic object, but we identify it with a Boolean function in the obvious way. Its depth is the maximum path length (in terms of number of variables along the path; i.e., the constant DT has depth 0). Its size is the number of leaves. Given a Boolean function $f$, we write $\text{DT}_{\text{depth}}(f)$ for the least depth of a DT that computes $f$.

Here is a trivial fact:
Fact 1.4. If $\text{DT}_{\text{depth}}(f) \leq d$ then $f$ has a DNF of width $d$ and also a CNF of width $f$.

Proof. Given the depth-$d$ DT for $f$, to get the DNF take the OR of all paths which lead to a 1-leaf. To get the CNF, note that there is clearly a depth-$d$ DT for $\neg f$, so there is a width-$d$ DNF for $\neg f$. But by de Morgan, we can make the negation of a width-$d$ DNF into a width-$d$ CNF. \hfill \Box

Definition 1.5. Assume we are concerned with functions $f$ over $n$ Boolean variables $x_1, \ldots, x_n$. A restriction or partial assignment $\alpha$ means fixing some of the variables to 0 or 1, and leaving the remaining variables free. We also say that the free variables are set to “star”, $\star$. We write $\text{stars}(\alpha)$ for the set of coordinates which $\alpha$ leaves free. We also write $f|_\alpha$ for the restricted function $\{0,1\}^{\text{stars}(\alpha)} \rightarrow \{0,1\}$.

Definition 1.6. We write $R_s$ for the set of all restrictions $\alpha$ with exactly $s$ stars. (We suppress from the notation the dependence on $n$.)

1.2 Statement of the Switching Lemma

The Switching Lemma is concerned with how applying a random restriction simplifies a DNF $f$.

Definition 1.7. A random restriction with $s$ stars is just a uniformly random restriction $\alpha$ from $R_s$. Equivalently, the star set $S$ is chosen uniformly at random from the $\binom{n}{s}$ possibilities, and then the unstarred coordinates $[n] \setminus S$ are fixed uniformly at random from the $2^{n-s}$ possibilities.

The Switching Lemma says the following. Suppose you have a DNF $f$ with small width, $w$. Further, suppose you hit it with (i.e., apply to it) a random restriction $\alpha$ with extremely few stars; i.e., you fix almost all the coordinates randomly. Then $f|_\alpha$ likely has tiny DT-depth. Precisely:

Theorem 1.8. (Håstad’s Switching “Lemma”) Let $f$ be a DNF of width at most $w$ over $n$ variables. Let $\alpha$ be a random restriction with $s = \sigma n$ stars, where $\sigma \leq 1/5$. Then for each $d \geq 0$ (and $\leq s$),

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

Some comments:

1. The same theorem is of course true for CNFs.

2. Sometimes you will see different constants in there rather than 10; sometimes 7, sometimes 5. Håstad even managed to prove 4.16 or something. The point is, never mind; it only matters that it’s a constant.

3. Note that there is no dependence on the size of $f$.

4. Think of $\sigma$ as the fraction of coordinates that gets stars.

5. Further, think of $\sigma = 1/(1000w)$ (and perhaps think of $w = \log n$). In this case, the expected number of $\star$’s per term of the DNF is at most 1/100, much less than 1! So out of 1000 terms in the DNF, perhaps just one or so will pick up even a single $\star$. And further, note that even this term with a $\star$ will very likely be fixed to 0. So the restriction is really hitting the DNF extremely hard.

6. On the other hand, note that there will still be $n/(1000w)$ variables which get stars, and if $w = \log n$ this is $\Omega(n/\log n)$ variables which remain free.
1.3 Why does this help prove Parity not in \( \text{AC}^0 \)?

It’s fairly easy to get the size lower-bound for depth-\( k \) circuits computing Parity out of the Switching Lemma. We will do the precise calculations later, but here is the very rough idea. Given an \( \text{AC}^0 \) circuit, we know that a random restriction is very likely to severely simplify each DNF at the bottom two layers [[draw picture]], at least assuming it has small width. Specifically, it’s quite likely that each small-width DNF will simplify to a small-depth decision tree. But we know a small-depth decision tree can be computed by a small-width CNF. So we have “switched” all the small-width DNFs at the bottom two layers into small-width CNFs. This lets us merge two layers of AND gates, and we’ve shrunk the depth by 1. We then repeat, overall making \( k \) random restrictions.

This leads to two things: first, we can compute the final restriction subfunction by a small-depth decision tree. But also, since each random restriction leaves a decent fraction of variables unset, there is still a decent fraction of variables unset after \( k \) restrictions. But any restriction of Parity is either Parity or its negation! And Parity on \( m \) variables (or its negation) requires a maximal-depth decision tree, depth \( m \) (this is easy to check). This leads to a contradiction. Again, we’ll do the details later.

2 Proof of the Switching Lemma

There are many many letters in the proof; please refer frequently to the following table:

\[
\begin{align*}
\text{f} &= T_1 \lor T_2 \lor T_3 \lor \cdots: \text{a DNF} \\
n &\text{: total number of variables f is on} \\
w &\text{: the width of f (max size of a term)} \\
\sigma &\text{: fraction of stars in the random restriction} \\
s = \sigma n &\text{: number of stars in the random restriction} \\
\mathcal{R}_s &\text{: set of all restrictions; has cardinality } \left(\begin{array}{c} n \\ s \end{array}\right)2^{n-s} \\
d &\text{: goal DT-depth for the restriction of f} \\
\mathcal{B} &\text{: the set of bad restrictions } \beta \text{ (making DT}_{\text{depth}} (f|_{\beta}) > d) \\
\beta &\text{: a fixed bad restriction} \\
\pi &\text{: a restriction on } d \text{ variables such that } f_{\beta \pi} \text{ is still not constant} \\
&\text{(gotten from the canonical DT for } f_{|\beta})
\end{align*}
\]

As mentioned, the proof of the Switching Lemma is somewhat hard. We give a combinatorial proof due to Razborov which most people consider simpler than Håstad’s probabilistic proof. This proof uses a very unusual strategy which I haven’t seen in many (any?) other proofs.

2.1 Proof strategy.

Let \( \mathcal{B} \) be the set of all bad restrictions, where a restriction \( \beta \) is bad if \( \text{DT}_{\text{depth}} (f|_{\beta}) > d \). Our goal is to show

\[
\frac{|\mathcal{B}|}{|\mathcal{R}_s|} \leq (10\sigma w)^d.
\]

We do this in a bit of a strange way. We define an “encoding” \( \text{Enc}(\beta) \) of each bad restriction \( \beta \). This encoding will consist of a restriction \( \beta' \) which is the same as \( \beta \) with a few more variables fixed — i.e., a few fewer stars — plus a little auxiliary info:

\[
\text{Enc}(\beta) = \text{some } \beta' \in \mathcal{R}_{s-d} + \text{a little bit of auxiliary info}.
\]
We will then show that there is a “decoding” procedure $\text{Dec}$ which takes $\text{Enc}(\beta)$ and returns $\beta$. In other words, the encoding maps each bad restriction $\beta \in B$ to something unique; we can recover $\beta$ from the encoding. Thus we have an injective mapping

$$B \hookrightarrow R_{s-d} \times \text{(a small auxiliary set)}.$$ 

Forgetting about the auxiliary set, this shows that $B$ is small; it’s at most $|R_{s-d}|$. And how big is $|R_{s-d}|$? Or more pertinently, given (1), how big is it compared to $R_s$? Intuitively, $R_{s-d}$ is much smaller because the real killer, information-theoretically, in specifying a restriction is saying where the ⋆’s go. And in $R_{s-d}$, you have to say this for fewer ⋆’s.

We’re being rough for now, so let’s say that

$$|R_s| = \binom{n}{s} 2^{n-s} \approx \frac{n^s}{s!} 2^{n-s}.$$ 

And,

$$|R_{s-d}| = \binom{n}{s-d} 2^{n-(s-d)} \approx \frac{n^{s-d}}{(s-d)!} 2^{n-(s-d)}.$$ 

So

$$\frac{|R_{s-d}|}{|R_s|} \approx \frac{s! 2^d}{(s-d)! n^d} \approx \left(\frac{2^s}{n}\right)^d = (2\sigma)^d.$$ 

Great! This is even better than (1). What’s going to happen is that

“little bit of auxiliary info” = about $d \lg w$ bits

(where $\lg = \log_2$). Hence the small auxiliary set will be of size

$$\approx 2^{d \lg w} = u^d.$$ 

This will make the final upper bound on the size of the encoding set $(2\sigma)^d \cdot w^d = (2\sigma w)^d$.

Well, we’ve been a bit sloppy/casual, and in the end we collect a few extra factors of $2^d$; hence the final bound of $(10\sigma w)^d$ in (1).

### 2.2 A good start

Going straight for that encoding is a bit ambitious, so we will start a bit slow. Let $\beta$ be a bad restriction, so $\text{DT}_{\text{depth}}(f|\beta) > d$. Let’s think about the function $f|\beta$ a bit. We can imagine getting $f|\beta$ by applying $\beta$ to each term $T_1, T_2, T_3, \ldots$ of $f$. It will be quite important in the proof that we consider these terms as ordered.

What happens when a term $T$ is restricted by $\beta$? Since $\beta$ is a restriction with few stars, probably most literals in $T$ get fixed, and maybe a small number stay free (i.e., get ⋆’s). An important thing to remember, though, is that $\beta$ is bad, hence $f|\beta$ is not constantly 1 (else it would have a depth 0 decision tree!). Hence $\beta$ does not fix any terms $T_i$ to 1; otherwise, it would make the whole DNF $f$ constantly 1. On the other hand, $\beta$ will probably “kill” many terms — i.e., fix them to 0. This is because it just has to fix one literal the “wrong” way to kill the whole term. In the unlikely event that $\beta$ does not kill $T$, it leaves it a nontrivial term $T|\beta$ over the starred literals (of which
there are presumably few).

When we talk now about applying $\beta$ to $f$, we know that it kills most terms, fixes no terms to 1, and leaves a few terms alive, but on fewer variables. Let’s focus now on the first term (in the ordering $T_1, T_2, \ldots$) which $\beta$ does not kill. We’ll write $T_i$ for that term, and we’ll write $U_1 = (T_i)|_\beta$ for the restricted version of that term, which is a conjunction on starred variables. Say for example $U_1$ has 3 literals, and let’s assume for now that $d \geq 3$.

**Claim 2.1.** Since $\text{DT}_{\text{depth}}(f|_\beta) > d \geq 3$, there is some way to fix $x_3, x_4, x_9$ such that $f|_\beta$ is still undecided.

Let $\pi_1$ be, say, the lexicographically least assignment to $x_3, x_4, x_9$ such that $f|_\beta$ is still undecided. Given a partial assignment $\pi_1$ like this, we will write $\pi_1$ for the restriction formed from $\beta$ by additionally assigning according to $\pi_1$. Note that

$$\beta \pi_1 \in \mathcal{R}_{s-3},$$

because $\pi_1$ fixes 3 more variables.

You may notice that we’ve got an object that we were looking for into play; namely, a restriction with even fewer stars than $\beta$. For example, suppose we could somehow have

$$\text{Enc}(\beta) = \beta \pi_1.$$

That would be a good start because we know $|\mathcal{R}_{s-3}|/|\mathcal{R}_s| \approx \Theta(1/n^3)$. So we would have encoded $\beta$ by something from a set $1/n^3$ smaller than the ambient restriction set $\mathcal{R}_s$.

Only trouble is, it’s not clear at all how to recover/decode $\beta$ from $\beta \pi_1$. Note that we’re not allowed to treat $\beta \pi_1$ syntactically as being $(\beta, \pi_1)$; for our counting purposes, we just know it’s some restriction in $\mathcal{R}_{s-3}$ and we don’t know which is the “$\beta$” part and which is the $\pi_1$ part.

A first idea to get out of this is to use the auxiliary information; we might tack some bits onto the encoding which say which of the variable-fixings in $\beta \pi_1$ comes from $\pi_1$. This would indeed let us recover $\beta$. But unfortunately, we’d need something like $3 \log n$ bits to specify these three variables, leading to an extra encoding-size factor of $n^3$, which defeats the purpose of mapping into $\mathcal{R}_{s-3}$.

**The main trick:** Here is the trick. Let $\gamma_1$ denote the assignment to the living variables in $U_1$ which makes $U_1$ true (i.e., 1). In our example, this is $x_3 = 1, x_4 = 1, \overline{x}_9$. Now instead of encoding $\beta$ by $\beta \pi_1$, we’ll consider

$$\text{Enc}(\beta) = \gamma_1 \beta \pi_1.$$

This is similarly in $\mathcal{R}_{s-3}$. But the beauty of this idea is the following: the restriction $\beta \gamma_1$ “tells” us which term $U_1$ is! More precisely:

**Claim 2.2.** If we consider $f|_{\beta \gamma_1}$, then $T_i$ is the first (in the ordering) term which $\beta \gamma_1$ sets to 1.

This takes a tiny bit of thought: the point is that certainly we still have that $T_1, T_2, \ldots, T_{i_1-1}$ are all still fixed to 0 by $\beta \gamma_1$, since they are fixed to 0 by $\beta$. And then $T_{i_1}$ is indeed fixed to 1, because $\gamma_1$ fixes $U_1 = (T_{i_1})|_\beta$ to 1.

Because of this, we can almost decode $\beta$ from $\beta \gamma_1$. With $\beta \gamma_1$ we can identify $T_{i_1}$. Now, we still need to pull out $\gamma_1$ from $\beta \gamma_1$, but the point is:
Claim 2.3. We can specify the variables $\gamma_1$ is fixing in $\beta\gamma_1$ using only $3 \log w$ bits of auxiliary information, rather than $3 \log n$.

This is because we only need to specify which variables in the width-$w$ term $T_1$ are the ones $\gamma_1$ fixes.

All in all, we’ve shown that we can encode $\beta$ in a decodable way with an object from
$$R_{s-3} \times \{0,1\}^{3 \log w}.$$  
And from our previous calculations, we have
$$\frac{|R_{s-3} \times \{0,1\}^{3 \log w}|}{R_s} \approx (2\sigma w)^3.$$  
This is a pretty good start, except we only have a power of 3, whereas we wanted a power of $d$. To complete the proof we need to somehow “iterate” the above argument.

One more small trick that will actually be crucial:

Claim 2.4. By adding an additional 3 bits of auxiliary information, we can also “specify” $\pi_1$.

This is because $\pi_1$ fixes the same variables as $\gamma_1$ (i.e., $x_3$, $x_4$, and $x_9$), just in different ways. So we can use 3 extra bits of auxiliary information to specify how $\pi_1$ fixes these variables.

2.3 The full argument

We got a power of 3 in the above example because we supposed that in the first unkillable term of $f|\beta$, there were 3 unset variables. Then we took a fairly great loss by just using that $d > 3$, which implied there was some way to fix these variables to keep $f|\beta$ undetermined. We now improve this argument.

Recall we have a width-$w$ DNF $f$ with an ordered set of terms $T_1, T_2, T_3, \ldots$, along with some bad restriction $\beta$. This means that $\text{DT}_{\text{depth}}(f|\beta) > d$. We will define a canonical decision tree for $f|\beta$, denoted $C(f|\beta)$, and therefore we will be able to say that in particular the depth of $C(f|\beta)$ is greater than $d$. This definition is a bit finicky; one needs to pay attention to it carefully.

Definition 2.5. The canonical decision tree $C(f|\beta)$ is defined as follows: Take the first term $T_1$ in order which is not killed by $\beta$. Say it reduces to the term $U_1$, on $d_1$ variables. Make a complete depth-$d_1$ decision tree over the variables in $U_1$ (querying them in order of their indices). Note that there will be exactly one path, call it $\gamma_1$, which forces $(T_1)|\beta$ to 1; we put a 1-leaf here. [Draw picture.] For all the other paths $\rho$, recursively tack on the canonical decision tree $C(f|\beta\rho)$. (If $f|\beta\rho$ is constant, of course just put that constant as a leaf.)

Note the slight intricacy here: We first fix in $\beta$, which kills a bunch of terms, and leaves some alive. We take the first living term and query all its variables. But now, having made each further assignment $\rho$ of $d_1$ variables, we may have that $\beta\rho$ kills many more terms which $\beta$ didn’t. Each subtree of the canonical decision tree here moves onto the first surviving term in $f|\beta\rho$.

Certainly $C(f|\beta)$ is a decision tree for $f|\beta$. So by assumption that $\beta$ is bad, it has depth exceeding $d$. Therefore we may define:
**Definition 2.6.** Let \( \pi \) be the lexicographically leftmost path of depth exceeding \( d \) in \( C(f|_B) \). Then trim it if necessary so it fixes exactly \( d \) variables. So we have \( \beta \pi \in R_{s-d} \) and is such that \( f|_{\beta \pi} \) is still undetermined; i.e., not a constant function.

As in the previous section, though, we won’t use \( \beta \pi \) in the encoding of \( \beta \). Rather, we will use assignments that lead to 1-leaves in \( C(f|_B) \).

**Definition 2.7.** Let \( T_1 \) be the first term not killed by \( \beta \), and let \( U_1 \) be its restriction under \( \beta \). Let \( d_1 \) be the number of variables in \( U_1 \). Let \( \gamma_1 \) be the setting to the variables in \( U_1 \) which makes it 1. On the other hand, let \( \pi_1 \) be the part of \( \pi \) which sets these variables. [[Draw Beame’s picture.]]

Assuming \( \pi_1 \) is not all of \( \pi \), continue the process. Note that in this case, \( \beta \pi_1 \) must kill \( U_1 \). Let \( T_2 \) be the first term not killed by \( \beta \pi_1 \), and let \( U_2 \) be its restriction under \( \beta \pi_1 \). Let \( d_2 \) be the number of variables in \( U_2 \). Let \( \gamma_2 \) be the setting to the variables in \( U_2 \) which makes it 1. On the other hand, let \( \pi_2 \) be the part of \( \pi \) which sets these variables. Keep going, until eventually \( \pi_\ell \) finishes all of \( \pi \). At this point, truncate \( \gamma_\ell \) to set just the variables that \( \pi_\ell \) sets.

Our encoding will now be:

\[
\text{Enc}(\beta) = \beta \gamma_1 \gamma_2 \cdots \gamma_\ell + \text{some auxiliary info.}
\]

Note that the restriction \( B = \beta \gamma_1 \gamma_2 \cdots \gamma_\ell \) here is in \( R_{s-d} \), which is what we’d like. Let’s see what auxiliary info we’ll need to decode this \( B \) back to \( \beta \).

First, as before we have that in \( f|_B \), the first term set to 1 is indeed \( T_1 \). Thus we can add \( d_1 \lg w \) bits of auxiliary information, specifying which variables in \( T_1 \) are the ones which \( \gamma_1 \) fixes. (Actually, since the Decoder doesn’t actually "know" \( d_1 \), we can have \( w+1 \) symbols, the \((w+1)st\) of which is a sentinel; so we actually need to use \( d_1 \lfloor \lg(w+1) \rfloor \leq d_1 \lg w + d_1 \) bits.) We also add an additional \( d_1 \) auxiliary bits to specify how \( \pi_1 \) sets these variables. Hence we’ve shown:

**Claim 2.8.** By adding at most \( d_1 \lg w + 2d_1 \) auxiliary info bits, the Decoder can determine \( T_1 \), \( \gamma_1 \), and \( \pi_1 \).

We can proceed with decoding. Since the Decoder knows \( \gamma_1 \) and \( \pi_1 \), it can consider the restriction \( B_2 := \beta \pi_1 \gamma_2 \cdots \gamma_\ell \). By construction, \( \beta \pi_1 \) kills all terms in \( f \) up to \( T_2 \), so the same is true of \( B_2 \). Also, \( \gamma_2 \) sets \( U_2 = (T_2)|_{\beta \pi_1} \) to 1, so the same is true of \( B_2 \). Hence:

**Claim 2.9.** \( T_2 \) is the first term in \( f \) fixed to 1 by \( B_2 \).

Hence the Decoder can determine \( T_2 \). So again:

**Claim 2.10.** By adding at most \( d_2 \lg w + 2d_2 \) auxiliary info bits, the Decoder can also determine \( T_2 \), \( \gamma_2 \), and \( \pi_2 \).

The Decoder can continue along, finding \( \gamma_3 \), \( \pi_3 \), etc. This proceeds until the Decoder has \( B_\ell := \beta \pi_1 \pi_2 \cdots \gamma_\ell \). The only difference now is that \( f|_{B_\ell} \) might have a “first term which is still undetermined”, rather than a “first term which is 1”. In any case, it can still use \( d_\ell \lg w + 2 \lg w \) auxiliary info bits to determine \( \gamma_\ell \). At this point the Decoder has completely determined the \( \gamma_1 \cdots \gamma_\ell \) part of the encoded restriction \( \text{Enc}(\beta) \) (it knows it’s done, since it knows this part fixes exactly \( d \) variables).

We conclude:

**Claim 2.11.** By using at most

\[
(d_1 \lg w + 2d_1) + (d_2 \lg w + 2d_2) + \cdots + (d_\ell \lg w + 2d_\ell) = d \lg w + 2d
\]

bits of auxiliary information, there is a Decoder which uniquely recovers \( \beta \) from \( B = \beta \gamma_1 \gamma_2 \cdots \gamma_\ell \).
2.4 Calculations

We’re now done except for calculations. We’ve shown that there is an injective mapping from the set $\mathcal{B}$ of bad restrictions into

$$R_{s-d} \times \{0, 1\}^{d \log w + 2d}.$$  

This set has cardinality

$$\binom{n}{s-d} 2^{n-(s-d)} \cdot (4w)^d.$$  

Since the cardinality of all $s$-star restrictions is $\binom{n}{s} 2^{n-s}$, we conclude the probability of getting a bad restriction is at most

$$\frac{\binom{n}{s-d} 2^{n-(s-d)} \cdot (4w)^d}{\binom{n}{s} 2^{n-s}} \leq \frac{s(s-1)(s-2)\ldots(s-d+1)}{(n-s+d)(n-s+d-1)\ldots(n-s+1)} (8w)^d \leq \left(\frac{s}{n-s+d}\right)^d (8w)^d \leq (10\sigma w)^d,$$  

where we used $\sigma \leq 1/5$ in the last step.

3 Lower bounds for Parity circuits

We now give the precise calculations showing that Parity requires depth-$k$ circuits of exponential size:

**Theorem 3.1.** (Håstad.) Assume $n \geq 2^{O(k)}$, where $k \geq 2$. Then computing Parity of $n$ bits by a depth-$k$ unbounded fan-in AND-OR circuit requires size $S \geq 2^{\Omega(n^{1/(k-1)})}$. In particular, for the circuit to be of polynomial-size it is necessary that $k \geq \Omega(\log n/\log \log n)$.

**Remark 3.2.** The implied constant can be made pretty good; I think Håstad can achieve .0718.

**Proof.** Suppose $C$ is any depth-$k$ circuit of size $S$ which computes Parity. It is an exercise to show that $C$ can be converted into a leveled depth-$k$ circuit, where the levels alternate AND and OR gates, the inputs wires are the $2n$ literals, and each gate has fan-out 1 (i.e., it’s a tree) — and the size increases to at most $(2kS)^2 \leq O(S^k)$. Since this only changes the constant in the $\Omega(\cdot)$ in the statement, we can assume the circuit is of this form. [[Draw picture.]]

Let’s first prove the theorem assuming:

**every gate at the bottom level has fan-in at most** $w := 20 \log S$.  

(2)

At the end we’ll see how to remove this assumption easily.

Assume without loss of generality that the bottom layer of $C$ is AND gates, so the bottom two layers consist of DNFs of width at most $w$. Suppose we apply a random restriction $\alpha_1$ to the circuit, with $\star$-fraction $\sigma = 1/(20w)$, and target DT-depth $d = w$. The Switching Lemma tells us that the probability a particular DNF fails, under restriction, to be representable by a depth-$w$ decision tree is at most

$$(10\sigma w)^w = (1/2)^w = (1/2)^{\log S \ll 1/S}.$$  

\footnote{Actually, this is a slightly tricky exercise. It’s easier if you only have to get size $O(S^k)$, which is almost the same for our purposes if you think of $k$ as a “constant”.

}
So by a union bound over all at most $S$ such DNFs, there is a positive probability (indeed, a high probability) that every DNF gets simplified to something representable by a depth-$w$ DT. But we know such functions are also representable by a width-$w$ CNF. If we now “plug in” these CNFs to the circuit, we can collapse layers 2 and 3 and get a new circuit, of depth $k - 1$, which has bottom fan-in at most $w$.

We can fix such a good restriction, and repeat. We apply restrictions $\alpha_2$, $\alpha_3$, $\ldots$, each with $\star$-fraction $\sigma = 1/(20w)$, and yielding depths $k - 2$, $k - 3$, etc. We do this $k - 2$ times, at which point we get down to a circuit of depth 2. At this point, the number of variables that are still $\star$ is

$$m := n \cdot \sigma^{k-2} = \frac{n}{(400 \log S)^{k-2}}.$$ 

But as we mentioned before, every restriction of the Parity function is either Parity (or its negation). And as we saw last class, Parity on $m$ variables requires DNF size $2^{m-1}$ and also CNF size $2^{m-1}$. Hence we better have:

$$S \geq 2^{m-1} \Rightarrow \log S \geq \Omega(m) \Rightarrow O(\log S)^{k-1} \geq n \Rightarrow \log S \geq \Omega(n^{1/(k-1)}),$$

which is what the theorem claims.

It remains to show how to remove the assumption (2). To do this, we simply initially hit the circuit with a random restriction $\alpha_0$ with $\star$-probability $1/100$. It’s easy to check that if this indeed reduces the bottom fan-in of $C$ to at most $w$ with positive probability, then we can run the rest of the argument and only lose a little more on the constant in the $\Omega(\cdot)$.

But this is straightforward. Suppose we have a bottom gate (and AND, say) with fan-in exceeding $w = 20 \log S$. A Chernoff-type bound shows that except with probability exponentially small in $w$ — and hence, $\ll 1/S$ — the gate gets at least, say, $(3/4)w$ non-$\star$’s. And each such non-$\star$ has a 1/2 chance of immediately killing this gate. Hence again, except with probability exponentially small in $w$ — hence $\ll 1/S$ — the gate gets killed. We can now union bound over all bottom-level gates to conclude that there is a high probability this initial restriction $\alpha_0$ kills all gates with width exceeding $w$. 

\qed