Notes for Lecture 23

Today we will conclude our discussion of parity lower bounds, and then we will start discussing proof complexity.

1 Håstad's Result

This section is devoted to a sketch of Håstad's tight result for the complexity of parity with bounded depth circuits. We begin by stating the key theorem:

Theorem 1. *(Switching Lemma) Suppose f is a k-CNF*[1] *over the variables $x_1, \ldots, x_n$. Pick at random a restriction that leaves a fraction p of the variables unfixed. For each of the $n(1-p)$ variables that are fixed, independently hardwire 0 or 1 as that variable's value. Then for every t,*

$$\mathbf{Pr}[\text{after the restriction } f \text{ can be expressed as a } t\text{-CNF and a } t\text{-DNF}] \geq 1 - (7pk)^t.$$

The theorem states that, after making the specified sort of restriction, we get a formula that can be expressed as a simpler DNF (if the original f was a CNF) or a simpler CNF (if the original f was a DNF) with exponentially high probability.

We state one more useful result before getting on to the analysis:

Lemma 2. *If f is a k-CNF (or a k-DNF) for $x_1 \oplus x_2 \oplus \cdots \oplus x_m$, then every term has size n.*

Now let C be a depth d circuit for parity of size S. Each level of C will consist entirely of AND gates or of OR gates, starting with AND gates at the top level. Each of the AND gates at the top level is a 1-CNF, each clause being a single variable. We can apply the Switching Lemma with $t = \log S$ and $p = 1/14$ so that each of these top-level gates becomes a $\log S$-DNF with probability at least $1 - 1/S$ (see Figure 1).

Since the AND gates of the top level have been switched to ORs, we can collapse the first two levels into one level of ORs to get a circuit that computes the parity of $n/14$ inputs such that the top gates have fan-in at most $\log S$ (see Figure 2).

Next we apply the Switching Lemma to the (new) top level of gates, each of which is a $\log S$-DNF (so $k = \log S$), with $p = 1/(14 \log S)$ and $t = \log S$ (see Figure 3). Thus each of these OR gates can be computed by a $\log S$-CNF with probability at least

$$\begin{aligned} 1 - \left(7\frac{\log S}{14 \log S}\right)^{\log S} &= 1 - \left(7\frac{1}{14}\right)^{\log S} \\ &= 1 - \frac{1}{2^{\log S}} \\ &= 1 - \frac{1}{S}, \end{aligned}$$

[1]We can take f to be a k-DNF as well by taking complements.

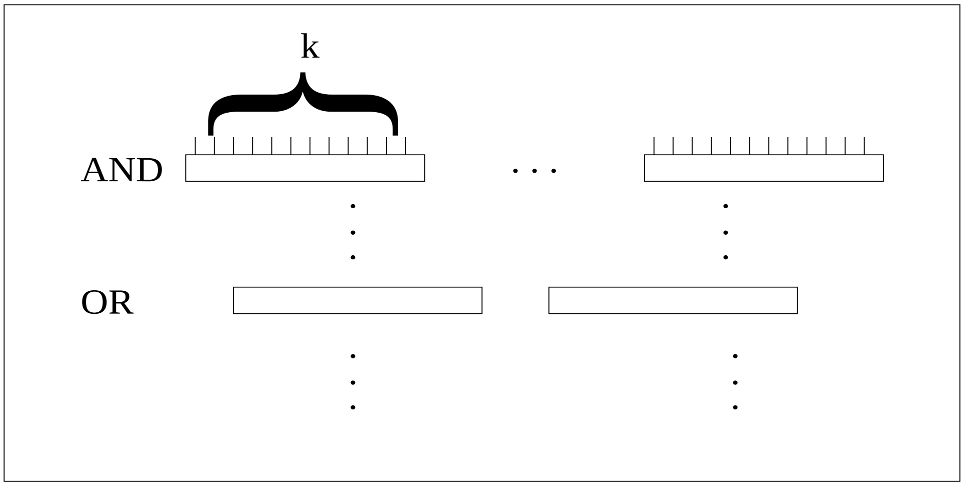

Figure 1: The first two levels of the circuit C. By the Switching Lemma, we can transform the AND gates of the top level to OR gates with high probability by making a random restriction.

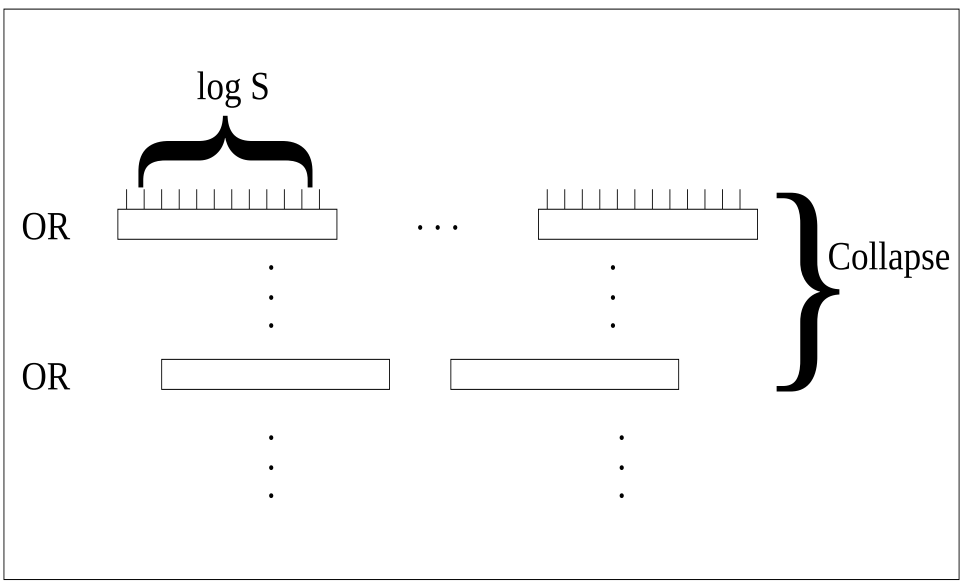

Figure 2: Applying the Switching Lemma to the AND gates transforms them to OR gates, so we can collapse the first two levels to form one level of ORs.

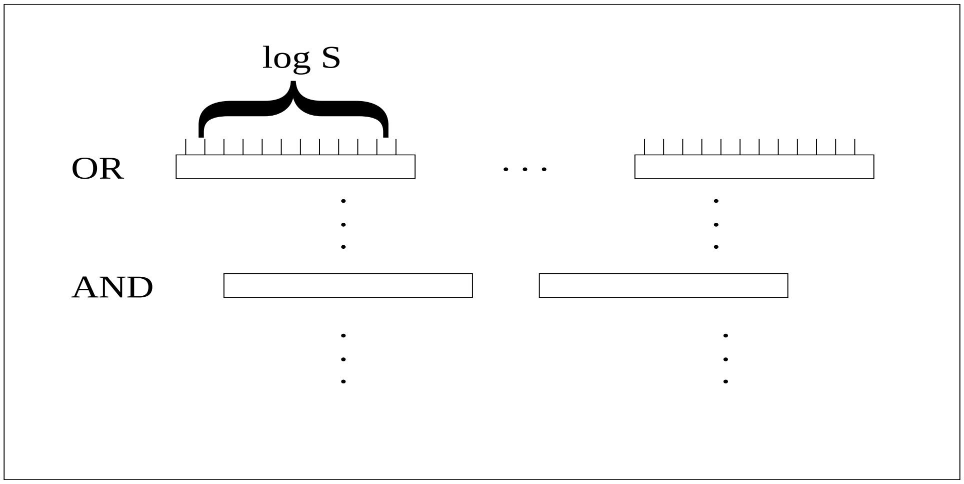

Figure 3: We use the Switching Lemma to transform the (new) first level of ORs to ANDs, and then we collapse the two consecutive levels of ANDs to one new level.

and after collapsing the two consecutive levels of ANDs, we get a depth $d-1$ circuit that computes the parity of $n/(14(14\log s))$ inputs such that the top gates have fan-in at most $\log S$.

Now we continue applying this procedure recursively until we get a depth 2 circuit that computes the parity of $n/(14(14\log S)^{d-2})$ inputs such that the top gates have fan-in at most $\log S$. By the lemma, the fan-in at the top must be at least the number of variables, so

$$\log S \geq \frac{n}{14(14\log S)^{d-2}}$$

or

$$S \geq 2^{\frac{1}{14}n^{1/d-1}}.$$

2 Proof Complexity

For the rest of this lecture and the next, we are going to discuss proof complexity, which is related to the question of **NP** vs. **coNP** in much the same way that circuit complexity is related to the question of **P** vs. **NP**.

Recall that **NP** = **coNP** if and only if there is some **coNP**-complete problem—the complement of an **NP**-complete problem—that is in **NP**. In particular, equality will hold if **UNSAT** is in **NP**, where **UNSAT** is the problem of determining if a Boolean formula φ is a contradiction—that is, if $\varphi(x_1,\ldots,x_n)=0$ for all truth assignments to the variables $x_1,\ldots,x_n$ of φ. If **UNSAT** is in **NP**, then we can make the following precise statements:

1. There is a polynomial-time verifier V and a polynomial bound p such that for all unsatisfiable formulas φ, there exists a proof π, with $|\pi| \leq p(|\varphi|)$, such that $V(\varphi,\pi)$ accepts, and

2. For all satisfiable formulas φ and all π, $V(\varphi, \pi)$ rejects (even if the length of π is not bounded by some polynomial in $|\varphi|$).

These statements follow immediately from the definition of **NP**.

For the purposes of studying proof complexity, we will consider a situation in which the requirement of a polynomial bound p in (1) is discarded. Thus, we say that V is a **sound and complete** proof system for **UNSAT** if

1. (**Soundness**) If φ is a satisfiable formula, then for all π, $V(\varphi, \pi)$ rejects,
2. (**Completeness**) If φ is an unsatisfiable formula, then there is a π such that $V(\varphi, \pi)$ accepts, and
3. (**Efficiency**) V runs in polynomial time.

We can now state a necessary and sufficient condition for **NP** to be different from **coNP** in terms of these sound and complete proof systems: **NP** $\neq$ **coNP** if and only if for every sound and complete proof system for **UNSAT**, there is an infinite sequence of unsatisfiable formulas whose shortest proof sizes grow superpolynomially.

What we are going to do now is to fix a natural proof system for **UNSAT** and then attempt to show unconditionally that there are formulas requiring superpolynomial proofs. This general approach can be likened to the approach physicists take when investigating physical theory: a physical theory offers some predictions about phenomena that can be observed in nature, and we check to see that those predictions are actually correct. If they are, we take that to be evidence in support of the theory. In this case, the "theory" is that **NP** $\neq$ **coNP**, and this theory predicts that for every proof system for **UNSAT**, there will be formulas requiring proofs of superpolynomial length.

Notice that any algorithm A for **SAT**, whether it runs in polynomial time or not, induces a proof system for **UNSAT**. The proof that φ is unsatisfiable is a description of a rejecting computation of A on input φ. Hence a lower bound on the proof length of a proof in this system automatically gives a lower bound on the running time of A.

2.1 Resolution Proof System

The proof system for **UNSAT** that we will investigate is Resolution. Given a CNF formula $\varphi = C_1 \wedge C_2 \ldots \wedge C_m$, where each C_i is a clause, a proof of the unsatisfiability of φ (also called a "refutation" of φ) is a sequence of clauses $(D_1, D_2, \ldots, D_t)$ such that

1. Each D_i is either
   (a) A clause of the original formula φ (i.e., C_j for some j between 1 and m) or
   (b) A clause obtained from D_j and D_h, $j < i$ and $h < i$, using the **Resolution rule**
$$\frac{x \vee D,\, \overline{x} \vee D'}{D \vee D'}, \quad \text{and}$$
2. D_t is $\underline{\mathbf{0}}$ (the false empty clause).

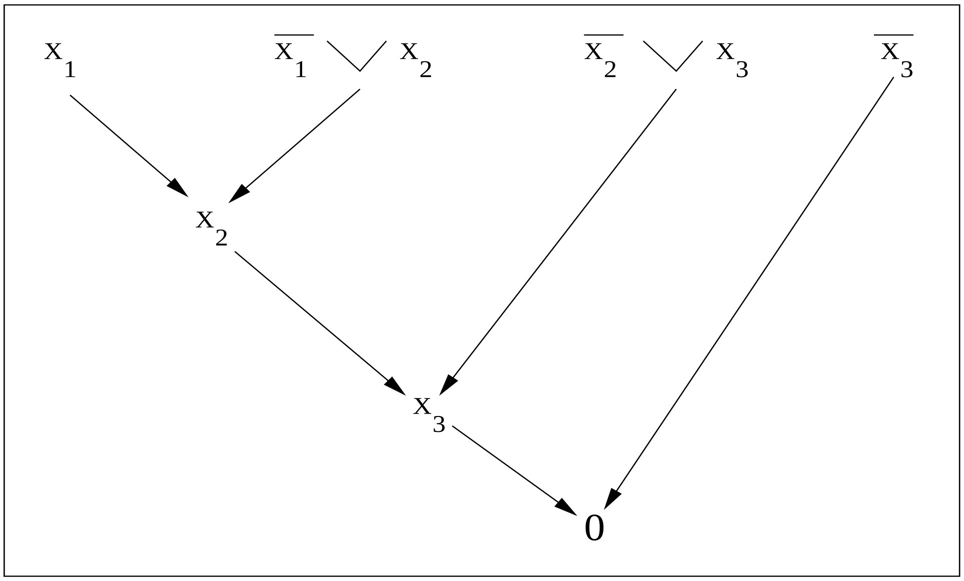

Figure 4: A tree-like Resolution proof.

As an example of how Resolution works, consider the formula

$$\varphi = x_1 \wedge (\overline{x_1} \vee x_2) \wedge (\overline{x_2} \vee x_3) \wedge \overline{x_3};$$

we can construct a proof as follows (see Figure 4):

1. Let $D_1 = x_1$, $D_2 = \overline{x_1} \vee x_2$, $D_3 = \overline{x_2} \vee x_3$, and $D_4 = \overline{x_3}$, so the first four clauses in the refutation are just the original clauses of φ.

2. From $D_1 = x_1$ and $D_2 = \overline{x_1} \vee x_2$, using the resolution rule we can deduce $D_5 = x_2$.

3. From $D_3 = \overline{x_2} \vee x_3$ and $D_5 = x_2$, using the resolution rule we can deduce $D_6 = x_3$.

4. From $D_4 = \overline{x_3}$ and $D_6 = x_3$, using the resolution rule we deduce the empty clause $D_7 = \underline{\mathbf{0}}$.

The proof that φ is unsatisfiable is then just the sequence of clauses $(D_1, D_2, \ldots, D_7)$. The reason that this is a proof is that the Resolution rule is sound: any assignment to the variables of φ that satisfies φ will also satisfy any clauses derived using the Resolution rule. Thus, if we derive a clause that cannot be satisfied by any assignment, it follows that φ can likewise not be satisfied by any assignment to its variables. This reasoning shows that the proof system is sound.

In this particular example, the proof can be pictured as a tree (again see Figure 4), so the Resolution proof for φ is called **tree-like**. In general, a proof can be represented as a directed acyclic graph, with the sequence $(D_1, D_2, \ldots, D_t)$ being a topological sort of that graph.

Now we want to show that Resolution is complete (i.e., every unsatisfiable formula has a refutation). In doing so, it will be useful to allow the use of the **weakening rule**

$$\frac{D}{D \vee x}.$$

We lose no generality in adding the weakening rule because, as it turns out, any proof that uses the rule can be converted to a proof that doesn't without adding to the proof's length:

Lemma 3. *If there is a proof for φ of length*[2] *k using the Resolution and weakening rules, then there is a proof for φ of length at most k using the Resolution rule only.*

We will not prove this result here.

Now let $\varphi = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ be an unsatisfiable formula over the n variables $x_1, x_2, \ldots, x_n$. Using the weakening rule, construct all clauses of length n that can be obtained from some C_i, $i = 1, 2, \ldots, m$, by adding literals. We claim that we can derive all length n clauses this way.

To see this, suppose that the clause

$$(x_1 = a_1) \vee (x_2 = a_2) \vee \ldots \vee (x_n = a_n) \qquad (*)$$

cannot be derived using weakening. Clearly the assignment

$$\left.\begin{array}{l} x_1 = 1 - a_1 \\ x_2 = 1 - a_2 \\ \ldots \\ x_n = 1 - a_n \end{array}\right\} \qquad (**)$$

contradicts $(*)$. Now take a clause $C_i \in \varphi$ over the variables $x_{i_1}, x_{i_2}, \ldots, x_{i_l}$, where each x_{i_j} is an x_k for some k. A clause is contradicted by exactly one assignment, so if C_i is contradicted by the restriction of $(**)$ to $x_{i_1}, x_{i_2}, \ldots, x_{i_l}$, it follows that C_i is a subset of $(*)$, but this is impossible since we assumed that $(*)$ cannot be derived from any C_j by adding literals. Hence $(**)$ satisfies every clause of φ, but this is a contradiction since we assumed that φ is unsatisfiable. (In other words, a length n clause that is not derivable using weakening describes a satisfying assignment for φ).

Having derived all length n clauses, we can use the Resolution rule to derive all length $n-1$ clauses. Continuing this process, we eventually derive the clauses x and $\overline{x}$ for a variable $x \in \{x_1, x_2, \ldots, x_n\}$, and one more application of the Resolution rule gives $\underline{\mathbf{0}}$, showing that φ has a refutation and that the proof system is indeed complete.

If $(D_1, D_2, \ldots, D_t)$ is a Resolution proof for φ, then t is the **length**[3] of the proof and the size of the shortest clause in the derivation is the **width** of φ (we only count clauses that are derived—not clauses that are in φ). It turns out to be easier to prove width lower bounds than length lower bounds directly. That this approach can still lead to strong lower bounds for length is the content of the following theorem:

Theorem 4. *Fix an unsatisfiable k-CNF formula φ over n variables. Let w be the minimal width of a proof for φ, let L_T be the length of the shortest tree-like Resolution proof, and let L be the length of the shortest general Resolution proof. Then*

$$\begin{aligned} w &\leq \log L_T + k \\ w &\leq O(\sqrt{n \log L}) + k. \end{aligned}$$

[2]The number of clauses in the proof

[3]We will also refer to the length of a proof as its **size** since the length is the number of nodes in the proof DAG.

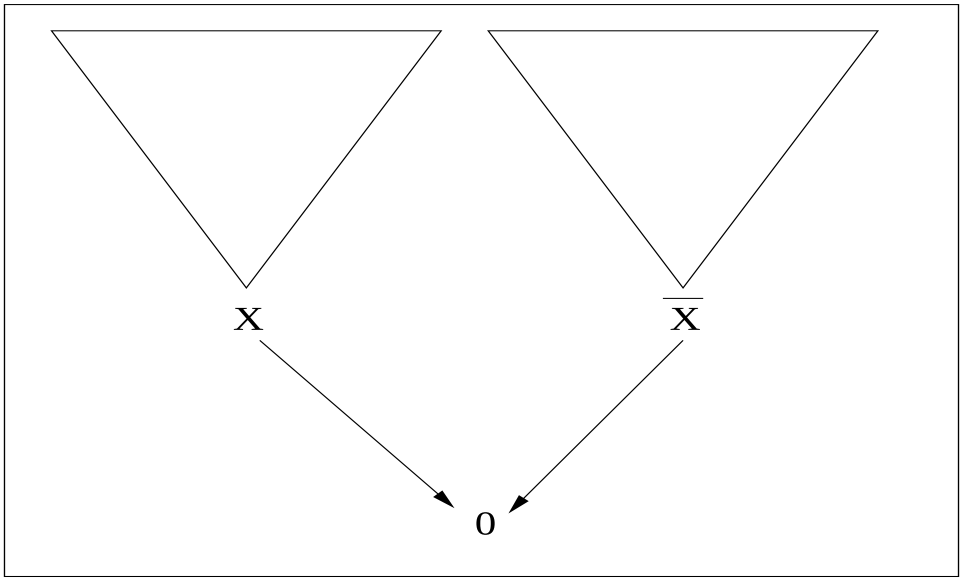

Figure 5: The tree-like Resolution proof of φ consists of two subtrees which are derivations of x and $\overline{x}$. Since the size of the proof is at most 2^b, one of these subtrees must have size at most 2^{b-1}.

We first state a lemma that will be useful in the proof of the theorem.

Lemma 5. *Let φ be as above, x a variable in φ, and $a \in \{0,1\}$. Suppose $\varphi_{x=a}$ is a k-CNF, the width of $\varphi_{x=1-a}$ is at most $d-1$, and the width of $\varphi_{x=a}$ is at most d. The φ has width no greater than* $\max d, k$.

We will prove the lemma and the second part of this theorem in the next lecture. For now, we focus on the first part of the theorem.

Proof. The strategy of the proof is to show that if φ has a short proof, then it has a proof of small width. We will show this by induction on n, the number of variables, and b, where b is an integer such that φ has a tree-like proof of size bounded above by 2^b. The claim is that if φ has a tree-like proof of size at most 2^b, then it has a proof of width at most $b+k$.

If $b = 0$, then the proof that φ is unsatisfiable is $(\underline{\mathbf{0}})$, in which case the claim is obvious.

If $n = 1$, then the only possible literals are x and $\overline{x}$, so any proof will have width 0 (or 1, depending on the conventions one chooses). Again the claim is immediate.

Now suppose φ has a tree-like proof of size 2^b. The last step of the proof must by a derivation of $\underline{\mathbf{0}}$ from x and $\overline{x}$, where x is a variable in φ (see Figure 5). Since the total size of the proof is 2^b, the tree-like derivation of x or the tree-like derivation of $\overline{x}$ must have size no greater than 2^{b-1}; without loss of generality, assume that the derivation of x is the small one.

Suppose that we substitute $x = 0$ into all the clauses of φ and all the clauses we derived from φ. The derivation is still correct since the Resolution rule still applies after we make this substitution, so the tree-like derivation of x becomes a derivation of 0, i.e. a proof for

$\varphi_{x=0}$. Hence $\varphi_{x=0}$ has a tree-like proof of size at most 2^{b-1}, and we can apply the inductive hypothesis to conclude that there is also a proof of $\varphi_{x=0}$ of width at most $b+k-1$.

Next make the substitution $x = 1$ into all the clauses of φ and all the clauses derived from φ. Then, as above, we get a proof of $\varphi_{x=1}$ of size at most 2^b. But since there is one less variable in $\varphi_{x=1}$, by induction on n there is a proof for $\varphi_{x=1}$ of width at most $b+k$.

The desired conclusion now follows from the lemma. $\square$