COMS 6998: Proof Complexity and Applications (Spring '25) February 6, 2025

Lecture 2: Resolution Lower Bounds Via The Pigeonhole Principle

Instructor: *Toniann Pitassi* Scribes: *Leo Orshansky*

1 Review: The Resolution Proof System

To start, we'll give a brief review of Resolution, one of the most classic and heavily-studied proof systems, which in the last lecture we proved to be sound and complete for refuting unsat CNF formulae. Today's lecture will be focused on proving exponential lower bounds in this proof system, but to start, let's remind ourselves of the definition:

Definition 1 (Resolution)**.** *A refutation in* Res *starts with a CNF formula $f = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, and derives additional clauses with the following rule:*

$$(A \vee x), (B \vee \bar{x}) \to (A \vee B)$$

where both items on the left hand side were either clauses in f, or clauses derived from previous applications of the Res *rule. A refutation is complete when the empty clause ϕ is derived. We sometimes refer to the* ***graph representation*** *of a refutation, which is the DAG formed by having one node per clause, and drawing each rule application with a directed edge from the parent clauses to the child clause. A refutation is called* ***tree-like*** *if each clause is used in at most one application of the rule (and as such, its graph representation is a tree).*

Observation 2. *A tree-like refutation of an unsat formula f corresponds to a decision-tree solution to Search_f, where for an assignment x, $\mathsf{Search}_f(x)$ outputs a clause in f which is violated by x.*

We will also define a game which plays out very similarly to the Res proof system, and can sometimes help us better capture the behavior of (non-necessarily-tree-like) Res refutations.

Definition 3 (Prover-Delayer Game)**.** *The prover-delayer game for formula f takes place over a series of rounds between a prover, P, and a delayer, D. P is trying to find a falsifying assignment to f, whereas D is trying to stretch the game to as many rounds as possible. In each round, the players do the following:*

1. *The prover chooses a variable x_i*
2. *The delayer chooses an assignment $b \in \{0, 1\}$, and tells the prover that $x_i = b$*
3. *The prover may declare that f is falsified, or may choose to "forget" the assignment of a variable, or may continue to the next round*

Observation 4. *There is a one-to-one correspondence between* Res *refutations of a formula f, and game graphs for the prover-delayer game on f. Tree-like proofs correspond to game trees, and non-tree-like structures correspond to prover strategies which are "forgetful". This correspondence is pictured in Figure 1.*

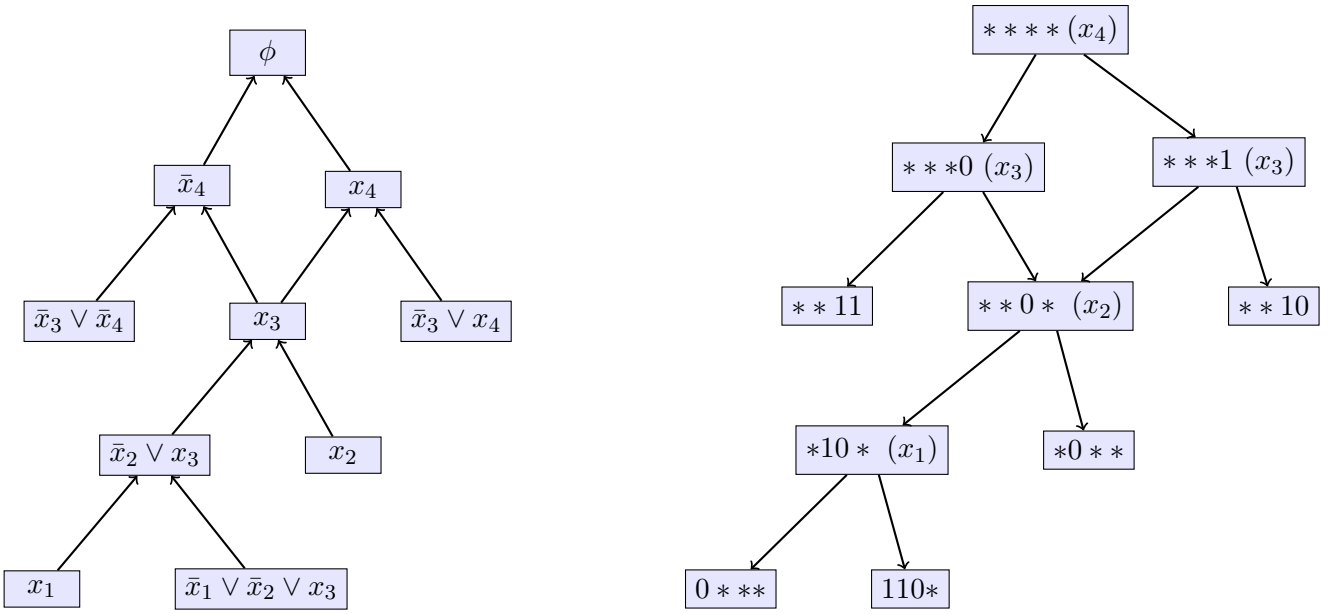

Figure 1: Res resolution (left) and prover-delayer game graph (right) for the unsat CNF formula $f = x_1 \wedge x_2 \wedge (\bar{x_1} \vee \bar{x_2} \vee x_3) \wedge (\bar{x_3} \vee x_4) \wedge (\bar{x_3} \vee \bar{x_4})$. The diamond-shaped behavior near the top of the graph represents the prover forgetting the value of x_4. Note that in each leaf, one of the clauses of the original formula is falsified.

2 Resolution Lower Bounds

We will now turn our focus to proving that Res does not contain short proofs for all unsat formulae – i.e. our goal is to find lower bounds on the size of a refutation for specific classes of formula. The class we will first focus on is that of Pigeonhole Principle statements (defined shortly below), and the primary technical step will be to lower bound the width of clauses in a refutation, which we define to be the number of literals in the clause. The analysis will be in two parts.

1. Proving that width lower bounds $\rightarrow$ size lower bounds: through a restriction argument, we will show that refutations with small size must also have small width throughout.

2. Proving width lower bounds: for Pigeonhole Principle refutations, we will show that there must be at least one wide clause by reasoning about the particular constraints and axioms of those formulae. We'll later take a look at a more general proof technique for width lower bounds in K-CNF refutations, which uses boundary expansion of clause-variable graphs.

2.1 Propositional Pigeonhole Principle

We will narrow our focus to a concrete family of unsat formulae on n variables, which we will call the Pigeonhole Principle statements.

Definition 5. *The Pigeonhole Principle statement with $n+1$ pigeons and n holes, denoted with* PHP_n^{n+1},

is the following CNF formula over variables $P_{i,j}$ (with $i \in [n+1]$ and $j \in [n]$):

$$\mathrm{PHP}_n^{n+1} = \underbrace{\bigwedge_{i=1}^{n+1} (P_{i,1} \vee \cdots \vee P_{i,n})}_{\textit{Pigeon clauses}} \wedge \underbrace{\bigwedge_{\substack{i_1 < i_2 \leq n+1 \\ j \leq n}} (\overline{P_{i_1,j}} \vee \overline{P_{i_2,j}})}_{\textit{Hole clauses (one-to-one)}} \wedge \underbrace{\bigwedge_{\substack{i \leq n+1 \\ j_1 < j_2 \leq n}} (\overline{P_{i,j_1}} \vee \overline{P_{i,j_2}})}_{\textit{Functional}} \wedge \underbrace{\bigwedge_{j=1}^{n} (P_{1,j} \vee \cdots \vee P_{n+1,j})}_{\textit{Onto}}$$

Here, each variable P_i, j represents the claim "pigeon i will sit in hole j", as pictured in Figure 2.

We'll briefly illustrate how this formula encodes the statement of the pigeonhole principle, section by section:

1. The pigeon clause for pigeon i is the OR of $P_{i,1} \ldots P_{i,n}$. This says, "pigeon i must have some hole".
2. The hole clause for hole j and a pair of pigeons i_1, i_2, is essentially the negation of $P_{i_1,j} \wedge P_{i_2,j}$, thus implying that those two pigeons cannot both sit in hole j. ANDed over all pairs of pigeons, this says that hole j can contain at most one pigeon.
3. The "functional" clause for pigeon i and a pair of holes j_1, j_2 is similar to a hole clause, but with the reverse statement: that the pigeon cannot sit in both holes at once. ANDed over all pairs of holes and all pigeons, this says that we have a mapping from pigeons to holes.
4. The "onto" clause for hole j says that at least one pigeon will sit in it. ANDed over all holes, (and combined with the previous), this says that we have a surjective mapping from pigeons to holes.

We note that the last two groups of clauses are optional, in the following sense. Clearly the formula is unsat just with the pigeon clauses and hole clauses by, well, the pigeonhole principle. Since the additional clauses are being ANDed with an unsat formula, any assignment that falsified the original formula will also falsify the combined formula, and so they are making it strictly "more unsat". However, there is also a sense in which they make it "less unsat" – if a pigeon can only be mapped to one hole, and every hole is mapped to by some pigeon, we guarantee that there is exactly one collision in the mapping, which seems to bring us closer to finding a (still nonexistent) injection from pigeons to holes. We will later see how this is useful for the proof.

Observation 6. *Any refutation for PHP_n^{n+1} (without the functional/onto clauses) is also a refutation for the full PHP_n^{n+1}. Thus, if $s(n)$-size refutations are possible for the former, then they are also possible for the latter.*

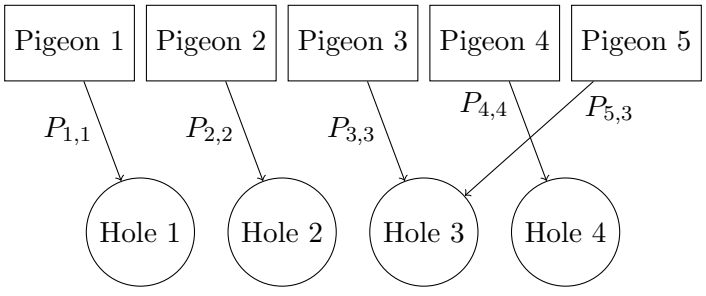

Figure 2: The pigeonhole principle, with corresponding literals in PHP_4^5 indicated

Corollary 7. *Proving $\Omega(s(n))$ lower bounds for the "raw" pigeonhole principle formula on $n+1$ pigeons, without the functional or onto clauses, reduces to proving $\Omega(s(n))$ lower bounds on PHP_n^{n+1}.*

2.2 Warm-Up: Tree-Like Res Lower Bounds for PHP_n^{n+1}

Before we go into proving that PHP_n^{n+1} requires exponentially large Res refutations, we will show a much simpler proof of the fact that this is true for the sub-class of tree-like Res refutations.

Actually, even before we do this, let's consider a naive upper bound for a tree-like refutation of PHP_n^{n+1}. Recall that tree-like refutations are in one-to-one correspondence with decision trees for Search_f, so let's try to construct a decision tree which solves $\mathsf{Search}_{\mathrm{PHP}_n^{n+1}}$. At the root of the tree, we'll start by querying for the location of pigeon 1 – if $P_{1,1}$ is true then we've found it in hole 1, but if not we ask for $P_{1,2}$, and so on until we've asked down to $P_{1,n}$, where if the answer is no then we can reject with the first pigeon clause. The key observation now is that at each location in the tree where we've "found" pigeon 1 in hole j, we can start solving the subproblem of placing pigeons $2 \dots n+1$ in holes $[n] \setminus \{j\}$ (an instance of PHP_{n-1}^n) – and recursing down all the way until PHP_1^2 where we reject on the corresponding hole clause. Since at each recursive level we have $O(n)$ nodes, and create $O(n)$ recursive subproblems, this decision tree has $O(n^n) = 2^{O(n \log n)}$ nodes.

Theorem 8. *Any decision tree solving $\mathsf{Search}_{\mathrm{PHP}_n^{n+1}}$ requires $2^{\Omega(n)}$ size.*

Proof. We will start by introducing a helpful subset of assignments, which we will call the *critical assignments*.

Definition 9 (Critical Assignments)**.** *For PHP_n^{n+1}, an assignment x to the variables $P_{i,j}$ is called* ***i-critical*** *if it satisfies all clauses besides the i^{th} pigeon clause. In other words, an i-critical assignment finds a hole for the n pigeons besides i, and leaves pigeon i unmapped. Note that there are $n!$ critical assignments for each i, so $(n+1)!$ in total.*

Lemma 10. *Any decision tree for $\mathsf{Search}_{\mathrm{PHP}_n^{n+1}}$ that gives correct answers for all critical assignments has size at least 2^n.*

Proof. Induction on n. For $n = 1$, clearly at least 2 nodes are required since the function is not constant (we could violate either one of the pigeon clauses). For $n > 1$, assume that the claim is true for $n-1$. Then, consider what happens when we query the value of $P_{i,j}$ at the root of our decision tree T.

If we see a 1, then we know that pigeon i goes to hole j, and the subtree S_1 rooted at this node is now a decision tree for the problem PHP_{n-1}^n. Take a critical assignment x for PHP_{n-1}^n, and assume that $S_1(x)$ is an incorrect answer for the search problem. Then, x plus $P_{i,j} = 1$ is a critical assignment for PHP_n^{n+1}, and $T(x) = S_1(x)$ is once again wrong. So by the inductive hypothesis, S_1 must have size at least 2^{n-1} for T to be correct on all critical assignments.

If $P_{i,j}$ is instead taken to be zero, then we can imagine that pigeon i is mapped to k for an arbitrary $k \neq j$. We can then extend the exact same argument as in the case above, where a critical assignment on n pigeons plus $P_{i,k} = 1$ is also critical for PHP_n^{n+1}, to prove that S_0 must have size at least 2^{n-1} for T to be correct on all critical assignments.

T has two subtrees of size 2^{n-1}, so its size must be at least 2^n. □

The theorem follows immediately, since a solution to $\mathsf{Search}_{\mathrm{PHP}_n^{n+1}}$ must be correct on all assignments, and in particular, all critical assignments. □

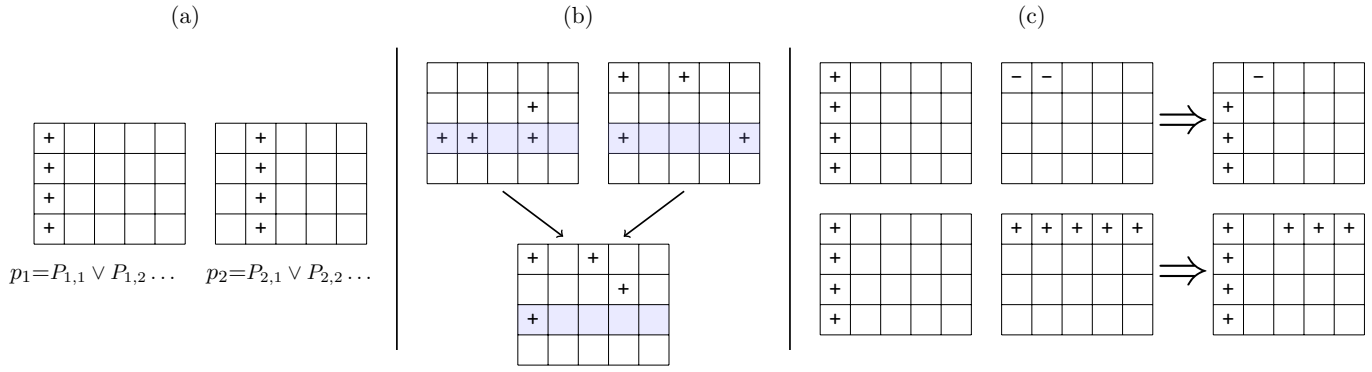

Figure 3: (a) Starting clauses of the monotone resolution refutation (pigeon clauses). (b) An example of the monotone resolution applied, in this case on the third row, or hole 3. (c) Example of a Res step transformed into a monotone resolution step. Note that on the bottom, we are not applying the monotone resolution rule – just translating the derived clause to a monotone clause.

Exercise 11. *Remove the gap between the upper and lower size bounds for tree-like refutations of* PHP_n^{n+1}.

2.3 Resolution Lower Bounds for PHP_n^{n+1} (The General Case)

In this subsection we will prove that *all* Res refutations for PHP_n^{n+1} require size exponential in n, not just tree-like ones as shown previously. This will require substantially more work, and so we'll split it into sections roughly as outlined at the beginning of Section 2.

2.3.1 Monotone Transformation of PHP_n^{n+1}

We'll start by transforming Res refutations of PHP into a nice combinatorial form. Once we prove that the transformed refutations are equivalent logically, and have the same asymptotic size, we can go on to achieve lower bounds against the size of Res refutations by proving lower bounds against the transformed, nicer version. All definitions are given below, but there is also a standard visual interpretation through matrices, which is pictured in Figure 3. In this representation, there is one matrix cell per variable $P_{i,j}$, and the row/column indices are flipped such that each row is a hole and each pigeon is a column. As such, there is one more column than row.

Definition 12 (Monotone Resolution for PHP_n^{n+1})**.** *A monotone resolution refutation of* PHP_n^{n+1} *starts with the* $n+1$ *pigeon clauses, which are all monotone. Additional clauses are derived using the following rule, which is parameterized by a particular hole* j*:*

$$C_1, C_2 \to \bigvee \left(\{P_{i,k} : P_{i,k} \in C_1 \cup C_2, k \neq j\} \cup \{P_{i,k} : P_{i,k} \in C_1 \cap C_2, k = j\} \right)$$

In simple terms, when the monotone resolution rule for hole j is applied to C_1 and C_2, the resulting disjunction contains all variables outside of hole j from either clause, but only contains variables for hole j if they were contained in C_1 and C_2. Intuitively, this rule encodes the fact that an assignment can satisfy C_1 and C_2 either by variables outside of hole j (in which case the resulting clause is satisfied by the inclusion of those variables), or by *the same* variable $P_{i,j}$ (by one-to-oneness of PHP_n^{n+1}).

Definition 13 (Clause Transformation from Res to Monotone)**.** *For a given clause* $c = \bigvee_{k=1}^{m} l_k$ *(where literals* l_k *are variables in* PHP_n^{n+1} *or their negations), the transformed monotone version of* c *is given as follows:*

$$M(c) = \bigvee \left(\{P_{i,j} : \exists k.\ l_k = P_{i,j}\} \cup \{P_{h,j} : \exists k.\ \exists i \neq h.\ l_k = \overline{P_{i,j}}\} \right)$$

In other words, $M(c)$ is the disjunction over:

- All positive variables $P_{i,j}$ in c
- For any negated variable $\overline{P_{i,j}}$ (stating that pigeon i does not go into hole j) in c, we add all other variables in the onto clause for hole j, to the disjunction. This essentially says "some pigeon that isn't i will go into hole j".

Claim 14. *Any size-*s Res *refutation* π *of* PHP_n^{n+1} *can be transformed into a monotone resolution refutation of size* $O(s)$*.*

Proof. A full proof will not be given, but a sketch is as follows: let's ignore the fact that the monotone-transformed initial clauses of π are not all legal starting clauses in the monotone resolution system (only the pigeon clauses are allowed). Next, we want to show that a monotone-transformed clause is logically equivalent to its precursor, given the pigeonhole axioms. This can easily be done by inspection, as illustrated in a note above. Finally, we simply need to convince ourselves that for any rule application $C_1, C_2 \to C_3$ in π, $M(C_3)$ can be derived from $M(C_1)$ and $M(C_2)$ using the monotone resolution rule at most a constant number of times. Once this is done, and the problem of initial clauses is fixed, the claim will follow. □

Now that we have a size-preserving transformation, we now know that lower bounds on monotone refutations of PHP_n^{n+1} are sufficient to achieve our overall goal.

2.3.2 PHP Lower Bound for Monotone Refutations

Observation 15. *Loosely speaking, a reason that we should expect monotone refutations of* PHP_n^{n+1} *to be large is that, to get from the initial set of clauses to the empty clause, we need to reduce the number of holes represented from* n *to 0, but the monotone resolution rule only allows us to "clear" one hole at a time. Additionally, clearing the* k^{th} *hole would seem to require a large amount of clauses with* $k-1$ *holes cleared – this is because as each holes is cleared, the resulting clauses must each have at least one more pigeon represented than in the previous step. Once there are many pigeons per clause, more combinations of clauses from the previous layer are needed in order to clear all variables from hole* k*. Overall, we expect to need roughly* n *layers with* $O(\binom{n}{k})$ *clauses in the* k^{th} *layer, which is a total size exponential in* n*.*

Motivated by the above observation, we prove the formal lower bound.

Theorem 16. *Any monotone refutation of* PHP_n^{n+1} *must have size* $\exp(\Omega(n))$*.*

Proof. We will proceed in two parts: as mentioned in the beginning of Section 2, we first prove that short refutations must have narrow clauses, and then we prove that all refutations must have at least one wide clause.

Take π to be a monotone refutation of PHP_n^{n+1}.

Lemma 17. *If π has size $s < 2^{n/20}$, then there is a restriction ρ mapping a constant fraction of pigeons to holes (without violating any PHP axioms), such that $\pi\big|_\rho$ has no clauses of width greater than $n(n+1)/10$.*

Proof. Let $t = n(n+1)/10$, and call a clause *wide* if its width (number of literals) is at least t. We will iteratively construct a restriction ρ which eliminates all wide clauses from π, as follows:

Begin with ρ initially empty. If we pick a random variable $P_{i,j}$ from among the $\leq n(n+1)$ variables which are not yet restricted under ρ, then a given wide clause will contain that variable with at least $1/10$ probability. Let w be the number of wide clauses in $\pi\big|_\rho$. By linearity of expectation, the average number of wide clauses which contain $P_{i,j}$ is at least $w/10$. Take P_{i^*,j^*} to be a choice of variable which achieves this average, and add $P_{i^*,j^*} = 1$ to the restriction ρ. Additionally, add $P_{i^*,j} = 0$ for $j \neq j^*$, and $P_{i,j^*} = 0$ for $i \neq i^*$. Repeat this procedure a total of $\log_{10/9} s + 1$ times, and then output ρ.

In each iteration, w decreases by a factor of at least $9/10$. This means that after all iterations are complete, $w \leq w_0 \cdot (9/10)^{\log_{10/9} s + 1} < w_0/s \leq 1$, thus all wide clauses have been eliminated. Additionally, the number of pigeons mapped by ρ is $\log_{10/9} < 1/3$. □

And now, to prove that π must have wide clauses:

Lemma 18. *π contains a clause of width greater than $2n^2/9$.*

Proof. Let the *complexity* of a clause $c \in \pi$ be the minimum number of pigeon clauses which jointly imply c on all critical assignments. Note that the complexity of each initial clause (a pigeon clause) is 1, whereas the complexity of the final clause (the empty clause ϕ) is $n + 1$.

We claim that for any monotone resolution rule application $C_1, C_2 \to C_3$, complexity$(C_3) \leq$ complexity$(C_1)+$ complexity(C_2). This follows from the fact that all assignments which satisfy C_1 and C_2 must also satisfy C_3, which is exactly the soundness property of the monotone resolution proof system. As such, the clause complexity at most doubles in each layer of the refutation, and we can in particular find some clause c^* such that $n/3 \leq$ complexity$(c^*) \leq 2n/3$. We now go on to show that c^* is wide.

Let $S \subseteq [n+1]$ be the minimal subset of pigeon clauses which implies c^*. As we know, $n/3 < |S| < 2n/3$. We now observe the following property which characterizes the set S:

- For any $i \in S$, there must be some i-critical assignment which falsifies c^*. This is because the i^{th} pigeon clause is violated *only* on the i-critical assignments (which concurrently satisfy all other pigeon clauses), and so $S \setminus \{i\}$ would be enough to imply c^* on all critical assignments unless there was a falsifying i-critical assignment.

- For any $i \notin S$, all i-critical assignments satisfy c^*. If not, then on the falsifying i-critical assignment, all pigeon clauses in S would be satisfied, and as such would not imply the truth value of c^*.

With this in mind, take α to be an i-critical assignment which falsifies c^*, for some $i \in S$. Take any $j \notin S$, and let $l \in [n]$ be the hole which pigeon j is mapped to under the assignment α. We observe that by modifying the values of just two variables in α, we can turn it into a j-critical assignment: we simply set $P_{j,l} = 0$ and $P_{i,l} = 1$. By the property above, this assignment must satisfy c^*, and since c^* is monotone, we know that $P_{i,l}$ must have been in the disjunction.

For a fixed i, we can repeat this reasoning for all $j \notin S$ to find a distinct variable $P_{i,l}$ in c^*, which is $n - |S|$ variables. But we can also vary i over all elements of S, and so we end up with a total of $|S|(n - |S|)$ distinct variables that must be in c^*. Since $2n/3 > |S| > n/3$, we get that c^* must contain at least $2n^2/9$ variables. □

Finally, we can combine these two lemmata in a very straightforward way. Since π has a clause of width $2n^2/9$, which is larger than $n(n+1)/10$, the contrapositive of Lemma 17 implies that π has total size $> 2^{n/20}$. In particular, it is clear that the size of π is $\exp(\Omega(n))$. □

Corollary 19 (Resolution Lower Bound)**.** *Any* Res *refutation of* PHP_n^{n+1} *has size* $\exp(\Omega(n))$.

Proof. Follows from Claim 14 and Theorem 16. □

3 Resolution Lower Bounds Beyond PHP

The main theorem proved in Section 2 showed us that monotone resolution (and, as a consequence, Res), is not capable of producing short refutations of the pigeonhole principle statement. Naturally, we would want to extend this to find more general classes of formulae for which Res does not have short refutations. Just like in the technical proof of the lower bound, we can break this into two parts: a reduction from size lower bounds to width lower bounds, and then the width lower bounds themselves.

3.1 General Size-Width Tradeoff

It turns out that general lower bounds can be proven on the size of Res refutations of a certain maximum width, with a stronger tradeoff for tree-like refutations. In particular, we have the following result due to Eli Ben-Sasson and Avi Wigderson.

Theorem 20 (BW01)**.** *Let F be an unsat k-CNF on n variables. Then,*

1. $\mathsf{Tree\text{-}Res\text{-}Size}(F) \geq 2^{\mathsf{Res\text{-}Width}(F)-k}$
2. $\mathsf{Res\text{-}Size}(F) \geq 2^{\Omega((\mathsf{Res\text{-}Width}(F)-k)^2/n)}$

Clearly, if we had a way to get general width lower bounds on Res refutations, we could combine it with this theorem to achieve very strong results. We will outline one such framework for width lower bounds.

3.2 Width Lower Bounds From (Boundary) Expansion

Definition 21 ((Boundary) Expansion Property of Clause-Variable Graphs)**.** *Take a bipartite graph $G = (V = (L, R), E)$. Take $|L| = m$ and $|R| = n$. We say that G is an (ε, δ)-expander (resp. boundary expander) if for all subsets $S \subseteq L$ such that $|S| \leq \varepsilon n$, $|N(S)| \geq \delta|S|$. Here, $N(S)$ is the set of nodes in $V \setminus S$ with at least one neighbor (resp. exactly one neighbor) in S.*

Lemma 22. *If G is a good expander with sufficiently low degree, then G is also a good boundary expander. In particular, if G is a (ε, δ)-expander with degree d, then it is a $(\varepsilon, 2\delta - d)$-boundary expander.*

Proof. Take a given left subset $|S| \leq \varepsilon n$. Let $E = \{v \in R : |N(v) \cap S| \geq 1\}$, and $B = \{v \in R : |N(v) \cap S| = 1\}$. By the expander property, $|E|/|S| \geq \delta$, and we will use this to show that $|B|/|S|$ is also large.

$$d|S| = \sum_{v \in R} |N(v) \cap S| = |B| + \sum_{\substack{v \in R \\ |N(v) \cap S| \geq 2}} |N(v) \cap S|$$

$$\geq |B| + 2|\{v \in R : |N(v) \cap S| \geq 2\}| \geq |B| + 2(\delta|S| - |B|)$$
$$|B|/|S| \geq 2\delta - d$$

□

We will now instantiate a particular kind of graph, clause-variable graphs of random k-CNF formulae, and use the expander property to give an informal argument for a width lower bound on their Res refutations.

Definition 23 (Random Clause-Variable Graphs)**.** *Let $\mathcal{F}(\Delta, n, k)$ be the distribution on boolean formulae taken as follows: out of n variables, pick $m = \Delta n$ random disjunction clauses of k literals each, and then output the formula $f = C_1 \wedge C_2 \wedge \cdots \wedge C_m$. Now let $\mathcal{G}(\Delta, n, k)$ be the distribution on bipartite graphs taken as follows: for each formula $f \in \text{support}(\mathcal{F})$, let $L = \{C_1, \ldots, C_m\}$ and $R = \{x_1, \ldots, x_n\}$, and add an edge from $C_i \to x_j$ if the i^{th} clause of f contains variable x_j (negated or unnegated).*

Claim 24. *For random graphs $G \sim \mathcal{G}(\Delta, n, k)$ (for appropriate choices of parameters), G is a boundary expander with high probability.*

Proof. Omitted. □

Claim 25. *For $\varepsilon > 1/4$, for a k-CNF formula f, if G_f is a $(\varepsilon, \Omega(1))$-boundary expander, then* Res *refutations of f must contain clauses with width $\Omega(n)$.*

Proof. Let π be Res refutation of f, and let c^* be the first clause in π for which the minimal set of initial clauses $C_1 \ldots C_m$ implying c^* has size at least $n/8$. Let $S \subseteq [m]$ be the set of indices of these clauses. By a complexity argument similar to the one in the proof of Lemma 18, we know that $|S| \leq n/4$. Now, take $B \subseteq [n]$ to be the index set of variables which are contained in exactly one clause of $\{C_i\}_{i \in S}$. For any x_j where $j \in B$, take $i \in S$ such that $x_j \in C_i$, and assume that $x_j, \overline{x_j} \notin c^*$. Since, by assumption, the clauses in $S \setminus \{i\}$ are not enough to imply c^* on their own, take an assignment α which satisfies every clause C_k for $k \in S \setminus \{i\}$, but falsifies C_i and c^*. If we flip the value of $\alpha(x_j)$, then c^* remains unsatisfied, and C_k for $k \in S \setminus \{i\}$ remain satisfied, since none of them contain x_j or $\overline{x_j}$ – but C_i is now satisfied, which is a contradiction since the set $\{C_i\}_{i \in S}$ must jointly imply c^*. Therefore, c^* must contain a literal for every variable on the boundary of its clauses, which, since G_f is a boundary expander and $|S| \leq n/4$, is $\geq \Omega(1) \cdot n/8 = \Omega(n)$ such variables. □

4 Aside: Res Upper Bounds for PHP_n^m

Since PHP_n^m is an "easier" formula to refute for $m \gg n$ than PHP_n^{n+1}, the exponential lower bound does not extend to such formulae. We will list the general landscape of known results:

1. For PHP_n^{n+1}: tree-like Res proofs are $2^{\Theta(n^2)}$, and general proofs are $2^{\Theta(n)}$ in size.
   - A similar Res lower bound exists for PHP_n^m, where $m = O(n^2)$
2. [BP97]: For $m \sim 2^{\sqrt{n}}$, there are polynomial-size Res refutations of PHP_n^m.
   - [Raz04]: This upper bound is nearly tight
3. [MPW02, PWW88]: In a different proof system, known as Res(polylog(n)), there are quasipolynomial-size refutations of PHP_n^{2n}.

5 Some Open Problems

1. Are there polynomial-size refutations of PHP_n^{2n} in the $\mathsf{Res}(\mathrm{polylog}(n))$ proof system? (Best known is quasipolynomial-size, i.e. $2^{\mathrm{polylog}(n)}$)

2. Are there polynomial-size and bounded-depth refutations of the weak PHP (say, where $m \sim 2^{\sqrt{n}}$)?

6 References

1. The original lower bound for Resolution proofs of the pigeonhole principle is due to Armin Haken [Hak85]. The lower bound for Resolution proofs of the weak pigeonhole principle is due to Ran Raz [Raz04].

2. Resolution lower bounds for Tseitin came next [Urq87] and explicitly introduced expansion as a key underlying combinatorial property. Lower bounds for random SAT were first proved in [CS88], and further simplified and improved in [BKPS02].

3. Resolution upper bounds for Resolution proofs of the weak pigeonhole principle are due to Buss and Pitassi [BP97]. Bounded-depth Frege upper bounds were first proved by Paris, Wilkie and Woods [PWW88]; a different quasipolynomial-sized Res(polylogn) was proven by Maciel, Pitassi and Woods [MPW02].

References

[BKPS02] Paul Beame, Richard M. Karp, Toniann Pitassi, and Michael E. Saks. The efficiency of resolution and davis–putnam procedures. *SIAM J. Comput.*, 31(4):1048–1075, 2002.

[BP97] Samuel R. Buss and Toniann Pitassi. Resolution and the weak pigeonhole principle. In Mogens Nielsen and Wolfgang Thomas, editors, *Computer Science Logic, 11th International Workshop, CSL '97, Annual Conference of the EACSL, Aarhus, Denmark, August 23-29, 1997, Selected Papers*, volume 1414 of *Lecture Notes in Computer Science*, pages 149–156. Springer, 1997.

[CS88] Vasek Chvátal and Endre Szemerédi. Many hard examples for resolution. *J. ACM*, 35(4):759–768, 1988.

[Hak85] Armin Haken. The intractability of resolution. *Theor. Comput. Sci.*, 39:297–308, 1985.

[MPW02] Alexis Maciel, Toniann Pitassi, and Alan R. Woods. A new proof of the weak pigeonhole principle. *J. Comput. Syst. Sci.*, 64(4):843–872, 2002.

[PWW88] Jeff B. Paris, A. J. Wilkie, and Alan R. Woods. Provability of the pigeonhole principle and the existence of infinitely many primes. *J. Symb. Log.*, 53(4):1235–1244, 1988.

[Raz04] Ran Raz. Resolution lower bounds for the weak pigeonhole principle. *J. ACM*, 51(2):115–138, 2004.

[Urq87] Alasdair Urquhart. Hard examples for resolution. *J. ACM*, 34(1):209–219, 1987.