Boolean Satisfiability
Lecture 7: Exponential lower bounds for Resolution

Edward A. Hirsch*

May 19, 2026

Lecture 7

In this lecture we

- Give a formal definition to proof systems and see how Resolution conforms to it.
- Prove an exponential lower bound on the size of treelike resolution refutations for the propositional pigeonhole principle.
- Establish the size-width relation for daglike resolution.
- Prove an exponential lower bound on the size of daglike resolution refutations for Tseitin formulas.

Contents

1 A formal definition of a proof system — and how Resolution conforms to it **2**
1.1 Propositional proof systems 2
1.2 Resolution as a proof system 3

2 A lower bound on the size of treelike resolution refutations **4**
2.1 Warm-up: Decision tree depth lower bound 4
2.2 Decision tree size lower bound 5

3 A lower bounds on the size of daglike resolution refutations **6**
3.1 Size-width relation: Daglike version 6
3.2 An exponential lower bound on the size of resolution refutations for Tseitin formulas . . . 7

4 Takeaway **10**

Historical notes and further reading **10**

*Ariel University, http://edwardahirsch.github.io/edwardahirsch

1 A formal definition of a proof system — and how Resolution conforms to it

1.1 Propositional proof systems

The definition.

Definition 1. A **proof system** for a language $L \subseteq \{0,1\}^*$ is a deterministic polynomial-time algorithm V that **v**erifies proofs:

Completeness. $\forall x \in L\ \exists w \in \{0,1\}^*$ such that $V(x,w) =$ "yes".

Soundness. $\forall x \notin L\ \forall w \in \{0,1\}^*$ inevitably $V(x,w) =$ "no".

Note that V runs in time polynomial in $|x|+|w|$ and not just $|x|$.

What "theorem" do we prove (and V checks our proof)? We prove that $x \in L$.

Complexity issues. In proof complexity, we are interested in *the size (length) of the shortest proof.*

Note that if there is a polynomial-size proof w for every $x \in L$, then L conforms to the definition of $L \in$ **NP** ! (We then say that V defines a **polynomially bounded proof system** for L.) This is unsurprising for $L =$ **SAT**, where a valid proof (of satisfiability) of F is a satisfying assignment for its n variables:

Completeness. $\forall F \in$ **SAT** $\exists w \in \{0,1\}^n$ such that $F[w] = 1$.

Soundness. $\forall F \in$ **UNSAT** $\forall w \in \{0,1\}^n$ inevitably $F[w] = 0$.

On the opposite, we do not know any polynomially bounded proof system for **UNSAT** (systems for **UNSAT**, or equivalently to the set of Boolean tautologies **TAUT**, are called **propositional proof systems**). Indeed, a polynomially bounded proof systems for **UNSAT** exists if and only if **NP** = **co-NP**, which is considered very unlikely. On the other hand, we do not know how to prove **NP** $\neq$ **co-NP** (it would also prove that **P** $\neq$ **NP**!).

Steve Cook's (or Cook–Reckhow's) program for advancing proof complexity asks to prove exponential size lower bounds for stronger and stronger proof systems one by one. That is, we prove an exponential lower bounds for a series formulas F_n hard for one proof system, then consider a proof system that has short proofs for these formulas F_n and prove an exponential lower bound for some other formulas G_n. Of course, we won't reach **NP** $\neq$ **co-NP** this way, but at least this way we are developing new complexity theory methods.

1.2 Resolution as a proof system

Let us give a formal definition for Resolution as a proof system.

Definition 2. Given $F = \bigwedge_{i=1}^{m} C_i$ in CNF,
a **(daglike) resolution proof** of F is a sequence of clauses D_j (for $j = 1, \dots, s$) such that

- For every j,
  - either $D_j \in F$ (i.e., $D_j = C_i$ for some i) (sometimes C_i's are called **axioms**),
  - or D_j is a resolvent of two clauses D_a, D_b with $a, b < j$.
- $D_s = \emptyset$ (the empty clause, the contradiction).

One can verify such a proof in polynomial time (if one states i, a, b explicitly, it would be even easier, that's why I asked to state them in the previous homework), thus we indeed defined a proof system. Its soundness follows from the soundness of the resolution rule (if an assignment satisfies D_a, D_b, then it satisfies their resolvent as well). Its completeness follows from the equivalence between a particular case of this proof system (namely, treelike resolution) and decision trees that we proved earlier in the course (obviously, decision trees are complete: one can examine all 2^n possibilities).

The **proof size** is always formally the bit-size of the proof, but here it is polynomially equivalent to s, so we will be talking about the number of clauses for convenience.

Also sometimes it is more convenient to include the weakening rule

$$\frac{C}{C \vee D},$$

but one can always get rid of it without increasing the proof size.

Let us define the treelike resolution system formally as well.

Definition 3. Given $F = \bigwedge_{i=1}^{m} C_i$ in CNF, a **treelike resolution proof** of F is a **binary tree** with every node $v \in \{1, \dots, s\}$ labeled by a clause D_v such that

- every leaf is labeled by some $D_j \in F$ (i.e., $D_j = C_i$ for some i),
- for every internal node v,
  its clause D_v is the resolvent of two clauses D_a, D_b labeling its two children.
- the root is labeled by $\emptyset$ (the empty clause, the contradiction).

The **proof size** here is still polynomially equivalent to s, which is the number of nodes in this tree.

It is easy to see that treelike resolution is a particular case of (daglike) resolution: just list all the clauses labeling the nodes level-by-level, this way we get a valid resolution proof.

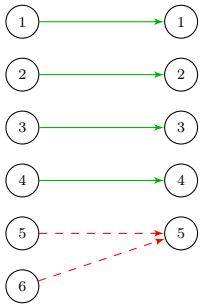
2 A lower bound on the size of treelike resolution refutations

We start with a simpler setting: treelike resolution. Recall that treelike resolution is size-equivalent to decision trees, so in fact we will be talking about decision trees.

We will prove lower bounds for the propositional pigeonhole principle. Recall that its negation can be expressed using the following clauses (denoted **PHP**) in the variables $x_{i,j} \sim$ expressing the statement "pigeon i sits in hole j (where $(1 \leqslant i \leqslant m$, $1 \leqslant j \leqslant m-1)$".

Pigeon-**H**ole **P**rinciple:

- **Pigeon axioms** (one for each pigeon $i \leqslant m$):
  $x_{i,1} \vee \ldots \vee x_{i,m-1}$: pigeon i is assigned some hole.
- **Hole axioms** (one for each hole $j \leqslant m-1$ and two different pigeons i, i'):
  $\overline{x_{i,j}} \vee \overline{x_{i',j}}$ $(\forall i, j, i' \neq i)$: pigeons i, i' do not share hole j.



As this formula states that there exists an injective mapping of m pigeons to $m-1$ holes, it is unsatisfiable.

2.1 Warm-up: Decision tree depth lower bound

Before we proceed to proving the size bound, let us consider an easier setting[1]: Let us prove a lower bound on the decision tree depth (that is, the length of the longest path from the root to a leaf).

Consider the following game between two players, called Prover and Oracle. They receive an unsatisfiable CNF formula $F(x_1, x_2, \ldots, x_n)$ on their input, maintain a partial assignment A to its variables (initially empty) and communicate by repeating the following protocol:

- Prover: chooses x_i that does not have any value in A and asks "What is the value of $\boldsymbol{x_i}$?"
- Oracle: answers by giving this variable its value v_i (now A is augmented by $\{\boldsymbol{x_i = v_i}\}$) and earns a coin

The game ends when the current assignment contradicts some clause of F (which will inevitably happen at some point, as F is unsatisfiable). Oracle's goal is to give values that will keep the game running as long as possible.

Let us formulate a straightforward lemma:

Lemma 1. *If there is a strategy for Oracle that guarantees her to earn at least d coins, then the depth of any decision tree for F is at least d.*

Using this lemma, we prove our lower bound:

Theorem 1. *The depth of any decision tree for **PHP** is at least $m-1$ (number of holes).*

[1] The author heard about this simple example from Dmitry Itsykson

Proof. Let us choose the following simple strategy for the Oracle: always return `False`. Note that hole axioms cannot be falsified this way as they contain only negative literals. Therefore, the game finishes when a pigeon axiom is falsified. Hoewever, pigeon axioms contain $m-1$ literals each, and therefore at least $m-1$ variables will get a value before falsifying such an axiom; thus Oracle will earn at least $m-1$ coins. By the above Lemma, the claim follows. □

2.2 Decision tree size lower bound

We now define a slightly more complicated game, relate it to the size of decision trees, and then apply this result to the propositional pigeonhole principle.

Prover–Delayer games. The game is now between Prover and Delayer (who still receive an unsatisfiable formula $F(x_1, x_2, \ldots, x_n)$ on their input, and who still maintain a partial assignment to its variables).

The communicate by repeating the following dialogue:

- Prover: chooses x_i that does not have any value in A and asks "What is the value of $\boldsymbol{x_i}$?"
- Delayer
  - either answers by giving this variable its value v_i (now A is augmented by $\{\boldsymbol{x_i = v_i}\}$) earning nothing,
  - or delegates it to Prover, in which case Delayer earns a coin and Prover chooses the value v_i and A is updated with $\{\boldsymbol{x_i = v_i}\}$.

The game ends when A contradicts some clause of F. Now Delayer is interested in delegating as many as possible decisions to Prover, (and through that also in running this game as long as possible because when the game ends, there is no chance to delegate or to do anything anymore).

Let us prove the following lemma relating successful Delayer's strategies to the size of decision trees.

Lemma 2. *If there is a strategy for Delayer that guarantees Delayer to earn at least d coins, then the size of any decision tree for F is at least 2^d.*

Proof. Given a decision tree T and such a strategy for Delayer, let us use it together with the following randomized strategy for Prover who uses T and starts in its root.

- Ask about the variable that is asked in the current node p of the decision tree.
- If Delayer delegates, choose a random value for $\boldsymbol{x_i}$ (with probability $1/2$ each).
- In T, go to the child of p according to the value of x_i (either obtained from Delayer or chosen at random as stated above).

Consider a specific leaf corresponding to a partial assignment A. How do we come to it?

- Delayer must choose the values consistent with A.
- Prover must pick the values consistent with A. It performs its random choice at least d times, as we assumed that Delayer has a strategy that brings at least d coins.

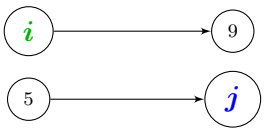
Therefore, the probability to finish the game in this leaf is at most $\mathbf{2^{-d}}$.

This applies, of course, to every leaf of T. On the other hand, the probability to finish in *some* leaf is 1. Therefore, there are at least $\mathbf{2^d}$ leaves in T. □

A lower bound on the size of decision trees for PHP. Let us consider the following Delayer strategy.

When Prover asks about $x_{i,j}$:

- `If pigeon` i `already has a hole`
  or
  `hole` j `is already occupied`
  `Then answer False`
  `Else delegate (and earn a coin` 🪙`)`



How many coins does Delayer earn?

- Hole axioms can never be falsified: if Delayer chooses a value, it is always `False`, and if Prover chooses a value, Delayer makes sure that it does not concern a hole that is already occupied.
- If a pigeon axiom $x_{i,1} \vee \ldots \vee x_{i,m-1}$ is falsified (that is, pigeon i becomes homeless), then, for every hole j, when Prover asked about $x_{i,j}$,
  - either Delayer delegated $x_{i,j}$ and earned a coin 🪙
  - or hole j was already occupied by a lucky pigeon i' — then who set $x_{i',j}$ to 1? Prover! Thus Delayer earned a coin at that moment 🪙

Thus for every hole, Delayer earned a coin, so this strategy always brings at least $m-1$ coins. Using Lemma 2 we thus conclude the proof of our lower bound:

Theorem 2. *Any decision tree for* ***PHP*** *must have* $2^{\Omega(m)}$ *leaves.*

3 A lower bounds on the size of daglike resolution refutations

3.1 Size-width relation: Daglike version

In Lecture 5 we proved the relation between the minimum size and width for treelike resolution refutations. We now prove it for the general case (daglike resolution refutations).

The strategy is the same: split our formula into two that have narrower proofs, then combine these proofs in one using the lemma that we proved in Lecture 5 (it is valid both for treelike and daglike proofs, as it concerns the proof width only).

Lemma 3 (see Lecture 5)**.** *Let F be in k-CNF.*
Let $F[\ell \leftarrow 0]$ be refutable within width $w-1$.
Let $F[\ell \leftarrow 1]$ be refutable within width w.
Then F is refutable within width $\max(w, k)$.

Theorem 3. *Assume we have a length-s daglike resolution refutation of a k-CNF F. Then F has a daglike resolution refutation of width at most $k + O(\sqrt{n \log_2 s})$.*

Proof. Set $\boldsymbol{d} := \sqrt{2n \ln s}$ and $\boldsymbol{a} := (1 - \frac{d}{2n})^{-1}$. We call a clause "fat" if it is wider than d. Our plan is to get rid of many fat clauses.

Let $\boldsymbol{f}$ be the number of **f**at clauses in our resolution refutation. We prove the following statement by induction on n (the number of variables) and b (defined below):

> "If a formula F has a resolution refutation containing less than a^b fat clauses, then F has a resolution refutation of width at most $k + b + d$".

The induction base ($b = 0$ or $n = 0$) is trivial. To prove the induction step, observe that, as there are only $2n$ literals, there must be a literal ℓ appearing in at least $\frac{d}{2n} \cdot f$ fat clauses of the refutation.

In $F[\ell \leftarrow \texttt{True}]$ they all disappear, thus there are at most $(1 - \frac{d}{2n})f < a^{b-1}$ fat clauses. Thus (induction on b) $F[\ell \leftarrow \texttt{True}]$ is refutable within width $k + b + d - 1$.

Concerning $F[\ell \leftarrow \texttt{False}]$, we use the induction hypothesis for $n - 1$ variables (as we get rid of ℓ) to conclude that $F[\ell \leftarrow \texttt{False}]$ is refutable within width $k + b + d$.

Combining these two refutations by Lemma 3 we get the induction hypothesis.

Now apply this statement for $b = \log_a s$ since the number fat clauses $\leqslant s - 1$. □

The following corollary is immediate:

Corollary 1. *If there are no width-$\boldsymbol{w}$ refutations of a k-CNF F, then all its proofs must have size* $2^{\Omega(\frac{(w-k)^2}{n})}$.

Note that this corollary is useful for the values of w that are much larger than both k and $\sqrt{n}$.

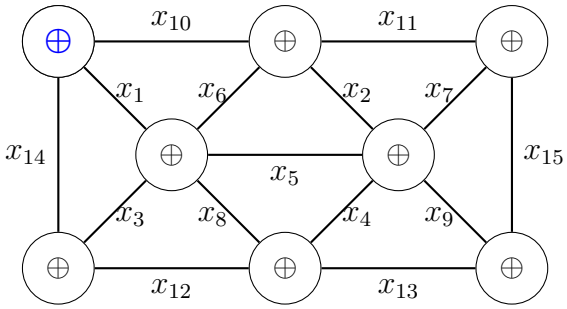
3.2 An exponential lower bound on the size of resolution refutations for Tseitin formulas

Tseitin formulas. Given a connected graph $G = (V, E)$, consider the following formula (called **Tseitin formula**). Introduce a variable x_e for every edge $e \in E$. For every vertex $v \in V$ of degree d_v, write a d_v-CNF expressing that $\bigoplus_{e \in v} x_e = 1$. Take the conjunction of all these CNFs.

Remark. 1. A Tseitin formula is unsatisfiable if and only if $|V|$ is odd.

2. One can label the vertices using 0's and 1's and consider equations $\bigoplus_{e \in v} x_e = c_v$, where c_v is the label of v. Then the unsatisfiability condition transforms into "$\oplus_{v \in V} c_v$ is odd". In this lecture we consider $c_v = 1$ for simplicity.

Example 1.



$$\begin{aligned} x_1 \oplus x_{10} \oplus x_{14} &= 1 \\ x_{10} \oplus x_6 \oplus x_2 \oplus x_{11} &= 1 \\ x_7 \oplus x_{15} \oplus x_{11} &= 1 \\ &\text{(etc.)} \end{aligned}$$

We will be interested in constant-degree graphs, so the size of the formulas is linear in the number of vertices.

Awareness function μ. We intend to introduce the "awareness function" $\mu\colon \{\text{clauses}\} \to \mathbb{R}_{\geqslant 0}$ such that

- $\mu(\text{axiom}) \leqslant 1$.
- $\mu(\texttt{False})$ is large.
- μ grows smoothly throughout the derivation.

Then we will be able to argue that there is a clause that has an "intermediate" value of μ.

A Boolean function f **implies** another Boolean function g if $\forall A \quad (f(A) = \texttt{True} \Rightarrow g(A) = \texttt{True})$. We say that g is **implied** by f Recall that we talked about "a clauses implied by a formula" in the context of CDCL algorithms.

For a clause C, define $\mu(C)$ as the minimum number of the input xor-equations implying C. (Note that while we are talking about CNFs, we can still talk about the xor-equation underlying the Tseitin formula we study.)

Obviously, for every input clause I, it must be that $\mu(I) = 1$ (it is implied exactly by the xor-equation of which it is a part of the CNF description).

It is easy to see (see homework HW#2) that that for a 2-connected[2] graph, no proper subset of our xor-equations is contradictory, that is, $\mu(\emptyset) = |V|$.

When we make a resolution step

$$\frac{A, \quad B}{C}$$

the resulting clauses C must be implied by the union of the sets of input xor-equations implying A and B, respectively. Thus $\mu(C) \leqslant \mu(A) + \mu(B)$.

It means that μ cannot increase more than twice, so in a resolution proof of the Tseitin formula for a graph (V, E) there must be a clause C^* such that

$$\frac{1}{3} \leqslant \frac{\mu(C^*)}{|V|} \leqslant \frac{2}{3}.$$

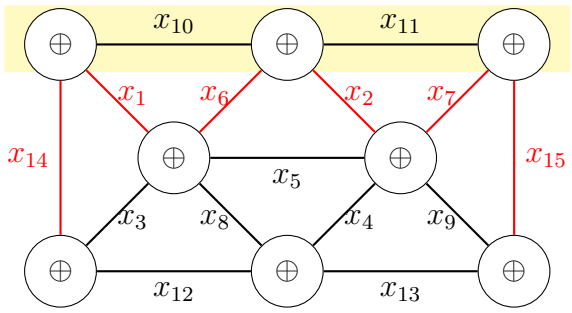
[2]A graph is called 2-connected if after removing a vertex it remains connected.

(Indeed, if there is no such clause, consider the first clause C in the refutation that has μ strictly greater than $\frac{2}{3}|V|$. Both its premises A and B have μ strictly less than $\frac{1}{3}|V|$, which is impossible as then $\frac{2}{3}|V| < \mu(C) \leqslant \mu(A) + \mu(B) < (\frac{1}{3} + \frac{1}{3})|V|$.)

Connecting width to expansion. Consider the Tseitin formula for a graph $G = (V, E)$. For a subset $U \subseteq V$ of vertices, define f_U as the set of the input xor equations for the vertices in U. We also identify this set with the Boolean function $\bigwedge_{u \in U} \left(\bigoplus_{e \ni u} x_v = 1\right)$ and the corresponding subformula of the Tseitin formula.

For a subset $U \subseteq V$ of vertices, define its edge **boundary** ∂U as the set of edges (and corresponding variables) leaving U. Note that for every variable x in this set, there is exactly one xor-equation $r = x \oplus \ldots$ in f_U where x occurs. Therefore, for every assignment A, if we flip the value of x in A, r is changed and the values of all other $r' \in f_U$ do not change.

Example 2.



Consider the conjunction of the three xor-equations corresponding to the three vertices in the yellow box:

$$\begin{aligned} x_1 \oplus x_{10} \oplus x_{14} &= 1 \\ x_{10} \oplus x_6 \oplus x_2 \oplus x_{11} &= 1 \\ x_7 \oplus x_{15} \oplus x_{11} &= 1 \end{aligned}$$

Boundary variables are colored in red.

It allows us to prove the following lemma.

Lemma 4. *Let $f_U \subseteq F$ be the minimum size subset of the input xor-equations implying a clause C. Then every boundary variable $x \in \partial U$ appears in C.*

Proof. Assume the contrary (that is, x does not appear in C). We know that $f_U = Y \cup \{r\}$, where r is a xor equation containing x, and Y does not contain x.

We claim that Y implies C, so f_U is not the minimum size subset. Assume for a moment that this is not the case, that is, there is an assignment B failing this implication:

$$Y[B] = \texttt{True} \text{ and } C[B] = \texttt{False}.$$

Note that as C and Y do not contain x, also

$$Y[B^x] = \texttt{True} \text{ and } C[B^x] = \texttt{False}.$$

(Recall that B^x is the assignment B with the value of the variable x flipped.)

We know that $r[B] \neq r[B^x]$, so one of these two values is `True`, and the corresponding assignment B' fails also the implication of C by f_U as for it $f_U[B'] = C[B'] \wedge r[B'] = \texttt{True}$ and $C[B'] = \texttt{False}$. □

Expander graphs are those that have a large boundary for every subset of vertices up to a certain size. In particular, there exist such graphs of constant degree k and such that for every $U \subseteq V$ of cardinality $|U| \leqslant \frac{2}{3}|V|$, the boundary has size $\Omega(|U|)$. These graphs are very much connected, in particular, 2-connected. We do not study expander graphs in this course, so just take this for given.

Recall now that our resolution refutation contains a clause C^* such that $\frac{|V|}{3} \leqslant \mu(C^*) \leqslant \frac{2|V|}{3}$. Consider the minimum-size set F_* of input xors implying C^*. Its size is between $\frac{1}{3}|V|$ and $\frac{2}{3}|V|$, thus for the expander graph we use its boundary has size $\Omega(|U|) = \Omega(|V|)$. By Lemma 4, C^* contains $\Omega(|V|)$ variables. By plugging this width lower bound into Corollary 1 we get the resolution refutation size lower bound

$$2^{\Omega(\frac{(w-k)^2}{n})} = 2^{\frac{(\Omega(n)-k)^2}{n}} = 2^{\Omega(n)}.$$

4 Takeaway

We have demonstrated an exponential lower bound on the resolution proof size. As we know, CDCL algorithms cannot run faster than the size of such a proof. In the next lecture, we will study stronger proof systems which have shorter proofs for these hard examples.

Historical notes and further reading

Formal definitions for proof systems and polynomial simulations were suggested by Steve Cook and Robert Reckhow in [CR79, R75].

Our definitions are a little bit different, but they are equivalent for our purpose. The first superpolynomial bounds for regular resolution we proved by Gregori Tseitin in the 1960s. He used linear equations modulo two for a grid graph. The bounds for unrestricted resolution were proved in the 1980s. Armin Haken proved exponential bounds for the propositional pigeonhole principle and Alasdair Urquhart proved exponential lower bounds for Tseitin formulas on expander graphs. Just before that Ofer Gabber and Zvi Galil gave the first construction of appropriate expanders.

Prover-Delayer games were suggested by Pavel Pudlák and Russell Impagliazzo [PI]. A uniform exposition of resolution lower bounds through the size-width technique can be found in the paper by Eli Ben-Sasson and Avi Wigderson [BSW].

For in-depth treatment of proof complexity and further historical remarks on the topic we refer the reader to a recent book by Jan Krajíček [Kra19].

References

[BSW] Eli Ben-Sasson, Avi Wigderson:
Short proofs are narrow — resolution made simple.
J. ACM 48(2): 149-169 (2001)

[CR79] Stephen A. Cook and Robert A. Reckhow. The relative efficiency of propositional proof systems. *J. Symb. Log.*, 44(1):36–50, 1979.

[R75] Robert A. Reckhow. On the lengths of proofs in the propositional calculus. PhD Thesis. *University of Toronto*, 1975.
https://www.cs.toronto.edu/~sacook/homepage/reckhow_thesis.pdf

[PI] Pavel Pudlák, Russell Impagliazzo:
A lower bound for DLL algorithms for k-SAT (preliminary version).
Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, *SODA-2000*, pp. 128–136, 2000.

[Kra19] Jan Krajíček: *Proof complexity.*
Encyclopedia of Mathematics and Its Appplications, Vol.170, Cambridge University Press, Cambridge - New York - Melbourne, (2019), 530pp.
Draft version: https://www.karlin.mff.cuni.cz/~krajicek/prfdraft.html