DD2242 Seminars on TCS: Proof Complexity Sep 7, 2016

Lecture 2

Lecturer: Jakob Nordström *Scribe: Susanna F. de Rezende*

1 Introduction

In the first lecture of this course we gave a brief (and therefore biased) introduction to proof complexity, defined proof systems such as resolution, cutting planes, and polynomial calculus, and also presented some interesting formulas to reason about. Although we started out talking about about tautological formulas, by the end of the lecture we had agreed that we can just as well focus on certifying unsatisfiable formulas in conjunctive normal form (CNF), since any tautology in propositional logic can be converted to a CNF formula of roughly the same size such that this CNF formula is unsatisfiable if and only if the original formula is a tautology.

In what follows, let us write UNSAT to denote the set/language of (syntactically well-formed) CNF formulas that do not have any satisfying assignments. Unless otherwise specified, in this and following lectures F will denote a CNF formula, which is usually assumed to be unsatisfiable. The highlights from the first lecture can then be summed up as follows.

Definition 1.1. A *proof system* for UNSAT is a deterministic algorithm $\mathcal{P}(F, \pi)$ that runs in time polynomial in the size $|F| + |\pi|$ of the input and is such that:

- if $F \in$ UNSAT, then there exists a *proof*, or *refutation*, π such that $\mathcal{P}(F, \pi) = 1$;
- if $F \notin$ UNSAT, then for any purported proof π it holds that $\mathcal{P}(F, \pi) = 0$.

Definition 1.2. A proof system $\mathcal{P}$ is *polynomially bounded* if there exists a polynomial p such that for all $F \in$ UNSAT there is a proof π such that $|\pi| \leq p(|F|)$ and $\mathcal{P}(F, \pi) = 1$.

It is widely believed that no polynomially bounded proof systems for UNSAT exist, because this would imply $\mathsf{NP} = \mathsf{coNP}$. As we discussed in the first lecture the converse is also true in that if $\mathsf{NP} = \mathsf{coNP}$ then there exists a polynomially bounded proof system for UNSAT (with a standard NP verifier acting as the proof checker $\mathcal{P}$). Since $\mathsf{P} = \mathsf{NP}$ implies $\mathsf{NP} = \mathsf{coNP}$, we can conclude the following theorem.

Theorem 1.3 ([CR79]). *If there are no polynomially bounded proof systems for* UNSAT*, then* $\mathsf{P} \neq \mathsf{NP}$.

One conceivable approach to prove $\mathsf{P} \neq \mathsf{NP}$ is to prove superpolynomial lower bounds for UNSAT for stronger and stronger proof systems until we reach some "global understanding" that makes it possible to prove lower bounds for completely general proof systems for UNSAT. This is known in the proof complexity community as *Cook's program*.[1]

In this course we will try to give a bit of an overview what is known about Cook's program. This program has not been very successful if measured in terms of progress towards its (very ambitious) final goal, but it has generated a lot of beautiful (and sometimes mysterious) mathematics. Today we will see a first example of this in the form of an exponential lower bound on proof length for the *resolution* proof system. Resolution was introduced in [Bla37] and started being studied more in earnest in the context of satisfiablity algorithms in [DP60, DLL62, Rob65]. This proof system is arguably the most well-studied proof system in proof complexity and is (still) the basis of state-of-the-art satisfiability algorithm using a paradigm known as *conflict-driven clause learning (CDCL)* [BS97, MS99, MMZ+01].

[1] Actually, Cook's program was not proposed by Steve Cook, at least not according to Steve Cook, but this name is well established. Probably the first to refer to this approach towards proving $\mathsf{P} \neq \mathsf{NP}$ was Peter Clote in his PhD thesis.

2 Resolution

The resolution proof system starts with the clauses of an unsatisfiable CNF formula F and iteratively derives new clauses until an explicit contradiction is reached. Let us recall the formal definition of this proof system.

Definition 2.1 (Resolution proof system). A *resolution refutation* π of an unsatisfiable CNF formula F, which we will often denote $\pi : F \vdash \bot$, is a sequence of clauses $\pi = (D_1, D_2, \ldots, D_{L-1}, D_L)$ such that D_L is the empty clause containing no literals, denoted $\bot$, and each clause D_i is either

(a) an *axiom clause* $D_i \in F$, or

(b) a clause on the form $D_i = B \vee C$ derived from clauses $D_j = B \vee x$ and $D_k = C \vee \overline{x}$ for $j, k < i$ by the *resolution rule*

$$\frac{B \vee x \qquad C \vee \overline{x}}{B \vee C} \ , \tag{2.1}$$

where we say that $B \vee C$ is the *resolvent over* x of $B \vee x$ and $C \vee \overline{x}$.

Without loss of generality we will assume that all clauses we encounter in resolution derivations are nontrivial in that they do not contain both x and $\overline{x}$ for any variable x. It is not hard to show thatany trivial clauses can always just be removed from a resolution derivation.

An important property for any proof system for UNSAT is that it should be *sound* (i.e., should never be able to refute a formula that in fact is satisfiable) and *(implicationally) complete* (i.e., should be able to refute any unsatisfiable formula. Resolution is sound and implicationally complete.

Lemma 2.2. *The formula F is unsatisfiable if and only if there exists a resolution refutation of F.*

Proof sketch. ($\Leftarrow$) Suppose there exists a satisfying assignment α to F, i.e., such that for every axiom clause in F the assignment α satisfies some literal in it. By induction over $\pi = (D_1, D_2, \ldots, D_{L-1}, D_L)$ we conclude that α must satisfy some literal in every resolvent (this follows by a simple case analysis for the resolution rule (2.1)). But this is a contradiction since there is no literal to be satisfied in the empty clause $\bot$ (this is why the empty clause is just another way of denoting contradiction).

($\Rightarrow$) This direction is not hard, but requires an argument. It is left as an exercise. □

We can think of a resolution refutation as a list of clauses annotated with explanations for each clause how it was obtained. This is illustrated in Figure 1a for the example CNF formula

$$F = (x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{z} \vee w) \wedge (\overline{z} \vee \overline{w}) \ . \tag{2.2}$$

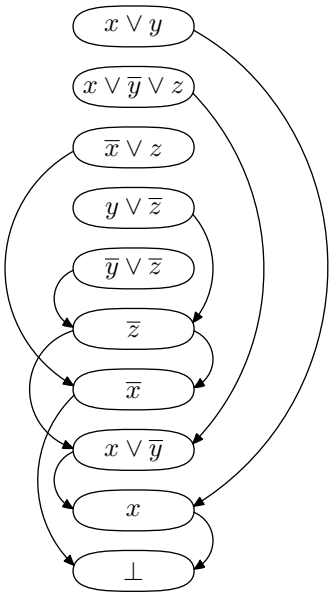
To every such refutation π we can also associate a DAG G_π in the following way. Sources of the DAG are axioms, and the unique sink is the empty clause $\bot$. Every node that is not a source has indegree two, and is the resolvent of its two predecessors. See Figure 1b for the proof DAG corresponding to the refutation in Figure 1a.

Given an unsatisfiable formula, the complexity measure we care most about is the *length* of refuting it in resolution. The length of a refutation π is the number of clauses in the refutation (so the length of the refutation in Figure 1a is 10), and is denoted $L(\pi)$. The length of refuting F is the length of a shortest refutation of F, and is denoted $L_{\mathcal{R}}(F \vdash \bot)$. We will sometimes drop the subscript and write just $L(F \vdash \bot)$ if it is clear from context that we are looking at resolution refutations.

Recall that as discussed in the first lecture we can also define the *size* of a refutation as the total number of symbols in it. When defining size in proof complexity we tend to be somewhat relaxed, however, in that we do not care too much about a linear factor more or less—what we are really interested in is whether the size is polynomial or superpolynomial. Since every clause can have at most linear size, the length and size measures for resolution differ by at most a linear factor, and in fact in the literature the size of a resolution refutation is usually defined to simply be the length. We will see other proof systems later in the course where the distinction between length and size is more important, however. In addition to length and size, other complexity measures of interest for resolution are, for instance, *clause space*, which is the number of clauses needed in memory while carrying out a refutation, and *width*, which is the size of the largest clause in the refutation, but in this lecture we will only care about length.

1.	$x \vee y$	Axiom
2.	$x \vee \overline{y} \vee z$	Axiom
3.	$\overline{x} \vee z$	Axiom
4.	$y \vee \overline{z}$	Axiom
5.	$\overline{y} \vee \overline{z}$	Axiom
6.	$\overline{z}$	Res$(4, 5)$
7.	$\overline{x}$	Res$(3, 6)$
8.	$x \vee \overline{y}$	Res$(2, 6)$
9.	x	Res$(1, 8)$
10.	$\bot$	Res$(7, 9)$

(a) Resolution refutation as an annotated list.



(b) Resolution refutation as a DAG.

Figure 1: Resolution refutation for the CNF formula in (2.2).

3 Pigeonhole Principle Formulas

The *pigeonhole principle* states that $n + 1$ pigeons do not fit into n pigeonholes if each pigeon needs its own hole. More formally, in its most basic version the pigeonhole principle states that there is no injective relation from $[n + 1]$ to $[n]$. In order to obtain an unsatisfiable formula we can encode (the negation of) this statement, i.e., that $n + 1$ pigeons do fit into n pigeonholes with at most one pigeon per hole, as a family of *pigeonhole principle (PHP) formulas* in conjunctive normal form.

We will define the PHP formulas for any number of pigeons $m > n$, although when m grows much larger than n it only makes the formulas easier to refute (which is intuitively clear, since the more pigeons we claim can be crammed into just n holes, the more obviously false the statement becomes). Once we have given the formal definition, for the rest of today we will fix $m = n + 1$.

For any positive integers m and n such that $m > n$ we define the formula PHP_n^m as follows. We have variables $x_{i,j}$, for $i \in [m], j \in [n]$, where we think of i as ranging over pigeons and j as ranging over pigeonholes. The intended interpretation of $x_{i,j}$ is that this variable is true if and only if pigeon i flies into hole j. We have clauses of two types:

Pigeon axioms For all $i \in [m]$ the clause

$$P^i = \bigvee_{j=1}^{n} x_{i,j} \tag{3.1a}$$

encodes that "pigeon i gets some hole."

Hole axioms For all $i, i' \in [m]$, $i < i'$, and all $j \in [n]$ the clause

$$H_j^{i,i'} = \overline{x}_{i,j} \vee \overline{x}_{i',j} \tag{3.1b}$$

encodes that "pigeons i and i' do not both fly to hole j."

The formula PHP_n^m is defined asas

$$PHP_n^m = \bigwedge_{i=1}^{m} P^i \wedge \bigwedge_{i'=2}^{m} \bigwedge_{i=1}^{i'-1} \bigwedge_{j=1}^{n} H_j^{i,i'} \quad . \tag{3.2}$$

The pigeonhole principle formulas are probably hands-down *the* most studied formula family in all of proof complexity. There is even a (very readable) survey paper [Raz02] dedicated exclusively to this family.

As discussed in the first lecture, PHP formulas come in several different flavours. If we want to think of pigeons flying into holes not as a relation but as a mapping, which is the way we usually reason about the pigeonhole principle, then we can add axiom clauses forbidding pigeons from going into multiple holes. And if we want to be fair to the holes, we can also require that every hole should get at least one pigeon. This yields the following additional types of clauses:

Functionality axioms For all $i \in [m]$ and all $j, j' \in [n]$, $j < j'$, the clause

$$F^i_{j,j'} = \overline{x}_{i,j} \vee \overline{x}_{i,j'} \tag{3.3a}$$

encoding that "pigeon i does not fly both to j and j'."

Onto axioms For all $j \in [n]$ the clause

$$S_j = \bigvee_{i=1}^{m} x_{i,j} \tag{3.3b}$$

encoding that "some pigeon flies to hole j."

By adding all clauses on the form (3.3a) and/or (3.3b) to the formula in (3.2), we obtain the *functional pigeonhole principle (FPHP)*, *onto pigeonhole principle*, or *onto functional pigeonhole principle (Onto-FPHP)*, respectively. Note that another way of describing Onto-FPHP formulas is that they claim that there is a perfect matching in a complete bipartite graph $K_{m,n}$ with m vertices on the left and n vertices on the right, and for this reason they are sometimes referred to as *perfect matching formulas.*

Despite the fact that PHP formulas encode such a basic combinatorial principle, and despite that they have been so heavily studied, there are still many (fascinating) open problems concerning these formulas in proof complexity. When it comes to the resolution proof system, however, PHP formulas are fairly well understood. Today, we will focus on the "vanilla version" (3.2) of the PHP formulas for $m = n + 1$ pigeons and prove that they are exponentially hard for resolution. This result, due to Haken [Hak85], is one of the celebrated early theorems in proof complexity.

Theorem 3.1 ([Hak85]). *Resolution refutations of* PHP^{n+1}_n *require length* $\exp(\Omega(n))$.

Note that PHP^{n+1}_n has $\Theta(n^2)$ variables, $\Theta(n^3)$ clauses, and size $\Theta(n^3)$. Therefore, in terms of formula size $N = \Theta(n^3)$ Theorem 3.1 yields a lower bound of $\exp\left(\Omega\left(\sqrt[3]{N}\right)\right)$.

It is possible to show (and it is not so hard) that resolution never needs length more than $\exp(\mathrm{O}(N))$, where N is the formula size. One might ask whether there exist formulas such that any refutation must have exponential length measured in terms of formula size or whether perhaps any formula of large enough size N can be refuted in length $\exp(\mathrm{o}(N))$. The answer is that the upper bound is the correct one. We will see in the next lecture formulas of size N with truly exponential lower bounds $\exp(\Omega(N))$ on resolution refutation length—i.e., tightly matching the worst-case $\exp(\mathrm{O}(N))$ upper bound up to constant factors in the exponent.

To prove Theorem 3.1 we need to show that there exists a $\delta > 0$ such that for large enough $n_0 \in \mathbb{N}^+$ it holds that if $n \geq n_0$, then any resolution refutation of PHP^{n+1}_n requires $2^{\delta n}$ steps. The challenge is that as $n \to \infty$ we get infinitely many possible refutations of PHP^{n+1}_n. We need to show that no such refutation can be shorter than $2^{\delta n}$. How can one prove such a thing? This is not obvious at all, and indeed Haken's paper [Hak85] is considered to be a major breakthrough. We will not quite follow Haken's paper, however, but will instead present a nicer and cleaner (and cuter) proof due to Pudlák [Pud00].

Pudlák presents the lower bound in terms of a two-person game. This game can be played over any unsatisfiable CNF formula F, but today we will define it only for PHP formulas.

4 Pudlák's Prosecutor–Defendant Game for PHP Formulas

Pudlák's game is played by two players: *Defendant*, who claims there is a way to fit $n+1$ pigeons into n holes; and *Prosecutor*, who wants to convict Defendant of lying. This should be a clear-cut case, but we have two problems. To begin with, this is a jury trial, and the jury will only be convinced by obviously, explicitly contradictory answers. On top of that, Prosecutor might need to leave early to pick up his kids at daycare (or stay home with them when they are ill), so he often needs a stand-in. The Prosecutor stand-in (whom we will refer to as Prosecutor from now on), who is not familiar with the details of this particular case, needs a book with super-explicit instructions how to cross-examine Defendant.

During the trial, Prosecutor will ask questions of the type "does pigeon i fly into pigeonhole j?" It is important to note that here i and j are not generic variables but always concrete numbers. Defendant answers "yes" or "no" to every question. Prosecutor then makes a note of Defendant's answer in a *record*. Unfortunately, Defendant can see Prosecutor's record and so always has full information about exactly what Prosecutor knows at each point in time, and can use this when choosing how to answer. For reasons that will become clearer later Prosecutor can also purposely forget some notes on the record. If so, Defendant will know that a previously given answer has been forgotten and can choose to answer differently next time the same question is asked.

Every record R consists of the questions-and-answers that Prosecutor remembers at a particular point during play, and looks something like

$$R = \{(i_1, j_1, \text{yes}),\ (i_2, j_2, \text{no}),\ (i_3, j_3, \text{no}),\ (i_4, j_4, \text{no}),\ (i_5, j_5, \text{yes}),\ \ldots\} \tag{4.1}$$

(for concrete numbers i_ℓ, j_ℓ). Prosecutor's goal is to convince the jury and, as we already mentioned, this requires an explicit contradiction. This means Prosecutor has to be able to produce a record R containing some contradiction of one of the following types:

1. (i_1, j, yes) and (i_2, j, yes), $i_1 \neq i_2$, both belong to R, i.e., Defendant says that two different pigeons fly into the same pigeonhole; or

2. $(i, 1, \text{no}), (i, 2, \text{no}), \ldots, (i, n, \text{no})$ all belong to R, i.e., Defendant says that a pigeon does not fly into any pigeonhole.

An *instruction book* (a.k.a. *Prosecutor strategy*) contains instructions what Prosecutor should do next based on the current record. For every record R that can be reached while Prosecutor plays according to this strategy, the book should contain a page with the record R together with an associated instruction what Prosecutor should do for his particular record. The instruction can be an *ask* instruction "ask whether pigeon i^* flies into hole j^*" or a *forget* instruction, for instance, "forget notes $\{(i_2, j_2, \text{no}), (i_5, j_5, \text{yes})\}$" (for the record in (4.1)). Observe that forget instructions can forget several notes, while ask instructions can only ask one question. It is important to note that Prosecutor can never forget *records* but only *notes* in records, and such a forget instruction creates another record, for which there must be another entry in the book with instructions of what to do.

As mentioned, an instruction book always has a page for every possible record that can arise during play. The record on the page must match exactly the current record (except the order of the notes is irrelevant). That is, if there is an entry in the book with a record R and an instruction to ask whether pigeon i^* flies into hole j^*, then there have to be two records $R \cup \{(i^*, j^*, \text{yes})\}$ and $R \cup \{(i^*, j^*, \text{no})\}$ taking care of the two possible answers from Defendant. We say that a Prosecutor strategy is *complete* if Prosecutor can win against any Defendant using the strategy. A strategy can be incomplete, however, in that Prosecutor gets stuck in infinite play if Defendant answers in particular ways. For instance, maybe Prosecutor chooses to forget information that later turns out to be crucial to force a contradiction.

Why would Prosecutor forget? The reason to forget is to reduce number of pages in the instruction book. It is not very impressive if Prosecutor needs to flip through several thousands of pages just to figure out which question to ask next. Prosecutor's objective is to minimize the size of a complete strategy/instruction book, i.e., the number of distinct records/pages in it.

5 Extracting a Prosecutor Strategy from a Resolution Refutation

Good, so the Pudlák game is clearly a cute game, but why are we doing this? The reason is that a short resolution refutation gives a complete Prosecutor strategy with few records. Hence, if we prove that no complete Prosecutor strategy can have few records, then we can conclude that there is no short resolution refutation.

Lemma 5.1. *If there exists a resolution refutation $\pi : PHP_n^{n+1} \vdash \bot$ of length $L(\pi) = L$, then there exists a complete Prosecutor strategy in the Prosecutor–Defendant game on PHP_n^{n+1} with $\mathrm{O}(L)$ records.*

Proof. Given a resolution refutation $\pi : PHP_n^{n+1} \vdash \bot$ in length $L(\pi) = L$, we will show how to construct a Prosecutor strategy with $\mathrm{O}(L)$ records.

Consider the DAG representation G_π of π where every vertex v is labelled by a clause in the refutation. Prosecutor constructs the strategy by walking from the sink, which is labelled by $\bot$, towards the sources. The contradictory empty clause $\bot$ was derived by resolving $x_{i,j}$ and $\overline{x}_{i,j}$ for some $x_{i,j}$, and Prosecutor will let this correspond to a record with the empty partial assignment together with an instruction to ask whether pigeon i flies to hole j. Prosecutor then moves to the clause/vertex falsified by Defendant's answer, i.e., if Defendant answers "yes" Prosecutor moves to vertex $\overline{x}_{i,j}$ and otherwise to $x_{i,j}$.

In general, Prosecutor will maintain the invariant that at the clause/vertex D_i in the refutation it holds that the corresponding record contains the minimal partial assignment falsifying D_i. Suppose that $D_i = B \vee C$ where this clause was derived by resolving the premises $B \vee x_{i,j}$ and $C \vee \overline{x}_{i,j}$ over x. Then the instruction associated to that record is to ask whether pigeon i flies to hole j and move to the clause falsified by Defendant's answer. In a second step, Prosecutor then forgets any notes/assignments that are not needed to falsify the premise clause. This clearly maintains the invariant.

In this way, Prosecutor can construct a strategy by processing all vertices in G_π, where every clause in π yields at most two records. Any sequence of answers from Defendant will yield a walk backwards through G_π. Sooner or later, Prosecutor will reach a source labelled by an axiom of PHP_n^{n+1}. By the invariant, this axiom clause is falsified by the record at that point in the game. If we look at the way explicit contradictions were defined above, it so happens that the record falsifying the axiom clauses constitutes an explicit contradictions (namely, a falsified hole axiom $H_j^{i_1,i_2}$ corresponds to a contradiction of type 1 and a falsified pigeon axiom P^i corresponds to a contradiction of type 2). The lemma follows. □

Example 5.2. Let us illustrate Lemma 5.1 by applying it to the CNF formula in (2.2). Strictly speaking, we cannot do this since we only defined the Prosecutor–Defendant game for PHP formulas, but the reader most likely has already realized that the question "does pigeon i fly to hole j?" is just a complicated way of saying that Prosecutor asks Defendant about the value of the variable $x_{i,j}$, and that "explicit contradictions" are just falsified axiom clauses. So let us show how the refutation in Figure 1 helps Prosecutor to win this generalized game played on the formula (2.2).

Prosecutor starts by asking about the value of x. Suppose Defendant answers 0, i.e., false. Then Prosecutor writes down $x = 0$ in the record, moves to the clause x, and asks about y. Defendant would immediately lose if answering 0, since this would falsify the axiom $x \vee y$, so suppose the answer is 1. This makes Prosecutor write down $y = 1$, move to the clause $x \vee \overline{y}$, and ask about z. Again, Defendant would lose immediately if answering 0, so let us assume the answer is 1. Given this answer, Prosecutor moves to the clause $\overline{z}$ and forgets the assignments $\{x = 0, y = 1\}$, keeping only $\{z = 1\}$. Finally, Prosecutor asks about y. Note that this question was already asked before, but since Prosecutor forgot the answer Defendant is free to answer in whichever way seems best right now independently of any previous answers. This does not help Defendant too much here, though, since if the answer is $y = 0$ Prosecutor obtains a contradiction to the axiom clause $y \vee \overline{z}$ and an answer $y = 1$ falsifies $\overline{y} \vee \overline{z}$.

6 Resolution Lower Bounds from Clever Defendant Strategies

In view of Lemma 5.1, all that is left to prove in order to obtain the lower bound in Theorem 3.1 is that any complete Prosecutor strategy for PHP_n^{n+1} must require a large number of records.

Lemma 6.1. *There is a $\delta > 0$ such that for large enough $n \in \mathbb{N}^+$ any Prosecutor instruction book for PHP_n^{n+1} must have at least $2^{\delta n}$ pages/records.*

To prove this lemma it is not enough to just consider a single round of play, as we did in Example 5.2, since Prosecutor can always win a game played on PHP_n^{n+1} by asking at most $n(n+1)$ questions to obtain a complete truth value assignment. Thus, we have to consider several rounds and prove that in order to always win, Prosecutor needs to be able to deal with exponentially many scenarios/records. In order to achieve this, we must devise clever Defendant strategies.

What might be a smart strategy for Defendant? We can start by observing that, intuitively, answering "yes" to a question "does pigeon i fly to hole j?" gives away lots of information (all there is to know about pigeon i), whereas answering "no" is not so informative (just one less whole where the pigeon could have gone, but there will typically be many more remaining). So perhaps answering "no" as often as possible might be a good Defendant strategy to attempt?

Defendant strategy 1. Defendant answers "no" to questions "does pigeon i fly to hole j?" as long as there is some other free hole for pigeon i and only answers "yes" when pigeon i has to go into hole j to avoid immediate contradiction.

Unfortunately, this idea does not work—it is not hard to see that Prosecutor has a strategy with just $\mathrm{O}(n)$ records to win in this case. (Note that this will not be a complete strategy, though.)

If we want to force Prosecutor to be able to deal with exponentially many different cases, one idea could be to use randomization to have Defendant play according to one of exponentially many different partial matchings of pigeons to holes.

Defendant strategy 2. Defendant chooses n out of $n+1$ pigeons randomly and matches pigeons to holes randomly, yielding a matching $\mathcal{M}$. Defendant then answers all questions consistently with $\mathcal{M}$ (i.e., responds "yes" to the question "does pigeon i fly to hole j?" if and only if i is matched to j in $\mathcal{M}$).

This makes Prosecutor's life slightly harder, but ultimately does not work either —Prosecutor has a counter-strategy with $\mathrm{O}(n^2)$ records. (We leave this as an exercise.) But what about mixing the two strategies?

Defendant strategy 3 (successful). Defendant chooses $n/4$ pigeons uniformly at random and assigns them to $n/4$ pigeonholes chosen uniformly at random to obtain a partial matching that we will denote $\mathcal{M}_{\text{init}}$. (We assume for simplicity that 4 divides n, and also that all divisions of n that will follow produce integers—this is without loss of generality since it is not hard to adjust the details to deal with this formally).

Let $\mathrm{dom}(\mathcal{M})$ denote the set of pigeons matched by a partial matching $\mathcal{M}$. Then $\mathcal{M}$ corresponds to a partial truth value assignment $\rho_{\mathcal{M}}$ in the natural way by letting

$$\rho_{\mathcal{M}}(x_{i,j}) = \begin{cases} 1 & \text{if } (i,j) \in \mathcal{M} \\ 0 & \text{if } (i,j') \in \mathcal{M} \text{ for } j' \neq j, \\ * & \text{if } i \notin \mathrm{dom}(\mathcal{M}) \text{ (where } * \text{ just denotes that the variable is left unassigned).} \end{cases} \tag{6.1}$$

We say that a partial matching $\mathcal{M}$ is *consistent* with a record R if for every note (i,j,yes) in R it holds that $\rho_{\mathcal{M}}(x_{i,j}) = 1$ and for every note (i,j,no) in R it holds that $\rho_{\mathcal{M}}(x_{i,j}) \neq 1$ (i.e., if $\rho_{\mathcal{M}}(x_{i,j}) = 0$ or $\rho_{\mathcal{M}}(x_{i,j}) = *$). Throughout the game, Defendant will maintain a partial matching $\mathcal{M}$ such that $\mathcal{M} \supseteq \mathcal{M}_{\text{init}}$ and $\rho_{\mathcal{M}}$ is consistent with the current Prosecutor record R (but in general $\mathcal{M}$ will contain extra information that is not known by Prosecutor).

It Prosecutor asks the question "does pigeon i fly to hole j?" Defendant answers "yes" if $(i,j) \in \mathcal{M}$ and "no" otherwise, and then considers whether to update the matching $\mathcal{M}$ as described next.

Let us say that record R *assigns* hole j to pigeon i if R contains the note (i,j,yes) and that it *prohibits* hole j for pigeon i if it contains the note (i,j,no). If for the pigeon i that Prosecutor just asked about it holds that $i \notin \mathrm{dom}(\mathcal{M})$ (in which case Defendant answered "no") then Defendant checks the number of prohibited holes for i in the current record. If this number is at least $n/2$, Defendant chooses

some hole j^* not prohibited for i and consistent with $\mathcal{M}$ (i.e., not occupied by any other pigeons) and sets $\mathcal{M} = \mathcal{M} \cup \{(i, j^*)\}$ (we can let j^* be the smallest index of such a hole if we want a concrete number). If it is not possible to update the partial matching $\mathcal{M}$ to accommodate pigeon i, then Defendant gives up.[2]

If Prosecutor forgets some notes in the record R to obtain R', then Defendant looks at every pigeon $i \in \mathrm{dom}(\mathcal{M}) \setminus \mathrm{dom}(\mathcal{M}_{\text{init}})$ such that some note $(i, j, \mathrm{yes}/\mathrm{no})$ was forgotten. If i does not have an assigned hole according to R' and the number of holes prohibited for i by R' is strictly less than $n/2$, then Defendant removes i from the matching $\mathcal{M}$. Note, however, that pigeons in the initial random matching $\mathcal{M}_{\text{init}}$ are never removed.

Let us say that a pigeon i is *thoroughly examined* in a record R if R assigns a hole to i or forbids at least $n/2$ holes for i, i.e., if R contains (i, j, yes) for some hole j or $(i, j_1, \mathrm{no}), (i, j_2, \mathrm{no}), \ldots, (i, j_\ell, \mathrm{no})$ for at least $n/2$ distinct holes j_ℓ. What this means is that Prosecutor has been forced to write down a lot of information about pigeon i. We claim that in order to win Prosecutor has to have lots of information about lots of pigeons on record simultaneously.

Lemma 6.2. *Before Defendant gets convicted when playing according to Strategy 3, Prosecutor must create a record R with $n/4$ thoroughly examined pigeons.*

Proof. As long as Defendant follows the strategy the answers are consistent with a partial matching of pigeons to holes, and hence cannot contradict any pigeon or hole constraints. This means that before conviction an update of $\mathcal{M}$ to accommodate some pigeon $i^* \notin \mathrm{dom}(\mathcal{M})$ must have failed. How did this happen? It must be the case that the current record R prohibits exactly $n/2$ holes for pigeon i^* but that there are no available free hole in Defendant's partial matching $\mathcal{M}$.

Since there are no free holes, all the $n/2$ holes that are not prohibited for pigeon i^* must be matched to another pigeon in $\mathcal{M}$. Thus, we have $|\mathrm{dom}(\mathcal{M})| \geq n/2$, which implies

$$|\mathrm{dom}(\mathcal{M}) \setminus \mathrm{dom}(\mathcal{M}_{\text{init}})| \geq |\mathrm{dom}(\mathcal{M})| - |\mathrm{dom}(\mathcal{M}_{\text{init}})| \geq n/4 \ . \tag{6.2}$$

But a pigeon i is in $\mathrm{dom}(\mathcal{M}) \setminus \mathrm{dom}(\mathcal{M}_{\text{init}})$ only if R either assigns a hole to pigeon i or prohibits $n/2$ holes for it, i.e., only if pigeon i is thoroughly examined. The lemma follows. □

We say that a record R with at least $n/4$ thoroughly examined pigeons is *informative*. By Lemma 6.2, for any choice of $\mathcal{M}_{\text{init}}$ a round of play passes through an informative record R. This record has to be consistent with $\mathcal{M}_{\text{init}}$, since Defendant always gives answers that are consistent with $\mathcal{M}_{\text{init}}$. We want to prove that any complete instruction book (i.e., one with which Prosecutor can win against any Defendant) must contain at least $2^{\delta n}$ distinct informative records.

We will prove the lower bound on the number of distinct informative records by showing that during a particular round of the game it is very unlikely that we see any fixed informative record.

Claim 6.3. For any arbitrary fixed informative record R it holds that

$$\Pr[R \text{ and } \mathcal{M}_{\text{init}} \text{ are consistent}] \leq 2^{-\delta n} \ ,$$

where the probability is over the random choice of an initial partial matching $\mathcal{M}_{\text{init}}$.

If we can prove Claim 6.3, then we are done. To see this, note that for any $\mathcal{M}_{\text{init}}$ that Defendant chooses, Prosecutor will build at least one informative record R before winning. In the first informative record, Defendant's inductive update of $\mathcal{M}$ has not yet failed, and so Defendant has answered consistently with $\mathcal{M}_{\text{init}}$ up to this point and R must be consistent with $\mathcal{M}_{\text{init}}$. This yields the sequence of inequalities

[2] Strictly speaking, we do not have any "give up" move for Defendant, but we could just define it as Defendant answering 0 to any question until the game is over, or whatever else fixed way of play that we like.

(where the probabilities are over $\mathcal{M}_{\text{init}}$):

$$\begin{aligned}
1 &= \Pr[\exists \text{ informative } R \text{ s.t. game passes through } R] && \text{[by Lemma 6.2]} \\
&\leq \sum_{\text{informative } R} \Pr[\text{game passes through } R] && \text{[by a union bound]} \\
&\leq \sum_{\text{informative } R} \Pr[R \text{ consistent with } \mathcal{M}_{\text{init}}] && \text{[by the reasoning above]} \\
&\leq 2^{-\delta n} \cdot (\#\text{ informative records } R) && \text{[by Claim 6.3]} \\
&\leq 2^{-\delta n} \cdot (\text{total } \#\text{ records } R) \ ,
\end{aligned} \tag{6.3}$$

which is just another way of saying that Prosecutor's instruction book must contain at least $2^{\delta n}$ records. So all that remains for us to do is to establish Claim 6.3.

Proof of Claim 6.3. Let $\mathcal{I}_R$ denote the set of thoroughly examined pigeons in R. For any fixed pigeon, the probability that it is chosen as one of the matched pigeons in $\mathcal{M}_{\text{init}}$ is $1/4$.[3] We know that $|\mathcal{I}_R| \geq n/4$ and thus the expected size of the intersection $\mathcal{I}_R \cap \text{dom}(\mathcal{M}_{\text{init}})$ is

$$\mathrm{E}[|\mathcal{I}_R \cap \text{dom}(\mathcal{M}_{\text{init}})|] = \sum_{i \in \mathcal{I}_R} \Pr[i \in \text{dom}(\mathcal{M}_{\text{init}})] \geq \frac{n}{16} \tag{6.4}$$

by linearity of expectation. Not only is the expected size of the intersection linear in n, but it can also be shown that the actual size of the intersection is extremely likely to be close to the expected value. By something that is called *concentration of measure*, and that we will talk a little bit more about below, it follows that except with exponentially small probability it holds that

$$|\mathcal{I}_R \cap \text{dom}(\mathcal{M}_{\text{init}})| \geq \frac{n}{32} \ . \tag{6.5}$$

For now, let us just accept that this is so and continue our line of argument.

Let $\mathcal{A}$ be the event "$\mathcal{M}_{\text{init}}$ and R are consistent" and let $\mathcal{B}$ be the event "$|\mathcal{I}_R \cap \text{dom}(\mathcal{M}_{\text{init}})| < \frac{n}{32}$". To prove the lower bound in Claim 6.3 we can use the trivial bound

$$\begin{aligned}
\Pr[\mathcal{A}] &= \Pr\left[\mathcal{A}\middle|\mathcal{B}\right] \cdot \Pr[\mathcal{B}] + \Pr\left[\mathcal{A}\middle|\overline{\mathcal{B}}\right] \cdot \Pr[\overline{\mathcal{B}}] \\
&\leq \Pr[\mathcal{B}] + \Pr\left[\mathcal{A}\middle|\overline{\mathcal{B}}\right]
\end{aligned} \tag{6.6}$$

from probability theory to deduce that

$$\begin{aligned}
&\Pr[\mathcal{M}_{\text{init}} \text{ and } R \text{ are consistent}] \leq \\
&\Pr\left[|\mathcal{I}_R \cap \text{dom}(\mathcal{M}_{\text{init}})| < \frac{n}{32}\right] + \Pr\left[\mathcal{M}_{\text{init}} \text{ and } R \text{ consistent} \,\middle|\, |\mathcal{I}_R \cap \text{dom}(\mathcal{M}_{\text{init}})| \geq \frac{n}{32}\right]
\end{aligned} \tag{6.7}$$

So it is enough to show that

$$\Pr[\mathcal{B}] = \Pr\left[|\mathcal{I}_R \cap \text{dom}(\mathcal{M}_{\text{init}})| < \frac{n}{32}\right] \leq 2^{-\delta' n} \tag{6.8}$$

for some $\delta' > 0$ and that

$$\Pr\left[\mathcal{A}\middle|\overline{\mathcal{B}}\right] \leq \Pr\left[\mathcal{M}_{\text{init}} \text{ and } R \text{ consistent} \,\middle|\, |\mathcal{I}_R \cap \text{dom}(\mathcal{M}_{\text{init}})| \geq \frac{n}{32}\right] \leq 2^{-\delta'' n}, \tag{6.9}$$

for some $\delta'' > 0$ to establish Claim 6.3 that $\Pr[\mathcal{M}_{\text{init}} \text{ and } R \text{ are consistent}] \leq 2^{-\delta' n} + 2^{-\delta'' n} \leq 2^{-\delta n}$, for some appropriately chosen $\delta > 0$ depending on δ' and δ'', from which Lemma 6.1 follows.

[3] If we want to be really picky, the probability is actually $\frac{n/4}{n+1} = 1/\left(4 + \frac{1}{n}\right)$, but again it is not hard to adjust the calculations that will follow to deal with this.

The probability in (6.8), which is the concentration of measure claimed above, can be estimated as

$$\begin{aligned}\Pr\left[|\mathcal{I}_R \cap \mathrm{dom}(\mathcal{M}_{\text{init}})| < \frac{n}{32}\right] &\leq \frac{\sum_{i=0}^{n/32-1}\binom{n/4}{i}\binom{n+1-n/4}{n/4-i}}{\binom{n+1}{n/4}} \\ &\leq \frac{\frac{n}{32}\binom{n/4}{n/32}\binom{3n/4+1}{7n/32}}{\binom{n+1}{n/4}} \\ &\leq 2^{-\delta' n}\end{aligned} \tag{6.10}$$

for some suitably chosen $\delta' > 0$, where the second-to-last inequality follows since the numerator is maximized for large i and the last inequality follows from a series of calculation using Stirling's formula

$$\sqrt{2\pi m}\left(\frac{m}{e}\right)^m e^{\frac{1}{12m+1}} < m! < \sqrt{2\pi m}\left(\frac{m}{e}\right)^m e^{\frac{1}{12m}} \tag{6.11}$$

which we will not perform here, because they are fairly standard and also quite tedious (but they are a useful exercise for readers who have not seen concentration of measure calculations before).

Instead of doing the formal calculations, let us give an intuitive argument why (6.8) holds. We obtain pigeons in $\mathcal{I}_R \cap \mathrm{dom}(\mathcal{M}_{\text{init}})$ by picking pigeons to include in $\mathrm{dom}(\mathcal{M}_{\text{init}})$ randomly one by one. Every time, the chance of hitting a pigeon in $\mathcal{I}_R$ is approximately $1/4$ and we do this $n/4$ times. If these events were independent, then they would be a series of coin flips of a biased coin, and it is well known that if we flip ℓ times a coin that has probability p of coming up heads, then the number of heads we actually see will be sharply concentrated around the expectation $p\ell$ (which in our experiment here is $n/16$). This is *not* a formal argument, since in our experiment we do not have independent events, but the dependencies are not too bad and so "morally" the argument above still works. However, the actual calculations to prove the inequality in (6.10) would just take too much time and effort for us to do at this point.

Let us instead proceed to prove the bound in (6.9), where we are giving a lower bound on the probability that $\mathcal{M}_{\text{init}}$ and R are consistent under the assumption that $|\mathcal{I}_R \cap \mathrm{dom}(\mathcal{M}_{\text{init}})| \geq n/32$. For every pigeon $i \in \mathcal{I}_R$ it holds that the record R either

1. assigns a hole j to pigeon i, or
2. prohibits $n/2$ distinct holes $j_1, j_2, \ldots, j_{n/2}$ for pigeon i.

In order for R and $\mathcal{M}_{\text{init}}$ to be consistent, the pigeons in $\mathcal{I}_R \cap \mathrm{dom}(\mathcal{M}_{\text{init}})$ must have been randomly matched in $\mathcal{M}_{\text{init}}$ consistently with these restrictions. Consider the pigeons in $\mathcal{I}_R \cap \mathrm{dom}(\mathcal{M}_{\text{init}})$ one by one. Let us say that a pigeon $i \in \mathrm{dom}(\mathcal{M}_{\text{init}})$ is *consistent* with R) if for the pair $(i,j) \in \mathcal{M}_{\text{init}}$ it holds that the note (i, j, yes) is consistent with R.

Let us first give a quick argument that conveys the intuition. Pigeons of type 1 have to hit exactly the right hole in $\mathcal{M}_{\text{init}}$, so the probability for such pigeons of being consistent is approximately $1/n \ll 1/2$. Pigeons of type 2 have to avoid half of the holes, so the probability of being consistent for them is at most $1/2$. Since the number of pigeons that have to be mapped consistently is at least $n/32$, we get a total probability of being correct of at most $(1/2)^{n/32}$ which is exponentially small. Done!

Well, actually we are not done, since unfortunately we do not have independence. Calculating more carefully (but not estimating very precisely, because we do not need to), we let the event $\mathcal{C}_i$ be "the ith pigeon in $\mathcal{I}_R \cap \mathrm{dom}(\mathcal{M}_{\text{init}})$ is mapped consistently with R by $\mathcal{M}_{\text{init}}$." We use another standard fact

$$\Pr\left[\bigcap_{i=1}^{\ell} \mathcal{C}_i\right] = \prod_{i=1}^{\ell} \Pr\left[\mathcal{C}_i \,\middle|\, \bigcap_{j=1}^{i-1} \mathcal{C}_j\right] \tag{6.12}$$

from probability theory and derive for the $(i+1)$st pigeon that

1. the probability that an $(i+1)$st pigeon of type 1 hits the right hole given that previous pigeons are consistent is at most $\frac{1}{n-i}$, and

2. the probability that an $(i+1)$st pigeon of type 2 avoids $n/2$ prohibited holes given that previous pigeons are consistent is at most $\frac{n/2}{n-i}$.

Since $i \leq n/4$, both probabilities above are less than $2/3$ and we can apply (6.12) to obtain that the probability that all pigeons are consistent is less than $(2/3)^{n/32} \leq 2^{\delta'' n}$ for some $\delta'' > 0$. This establishes the inequality in (6.9). As discussed above, we can now conclude the proof of Claim 6.3 by plugging (6.8) and (6.9) into (6.7). □

7 Recap of Proof

Just to review quickly what happened in this lecture, we proved exponential lower bounds on the length of resolution refutations of pigeonhole principle (PHP) formulas. We presented our proof in the language of Pudlák's Prosecutor–Defendant game. In this game Defendant claims to have a satisfying assignment for an unsatisfiable CNF formula F and Prosecutor interrogates Defendant about what values are given to the variables in F according to this assignment. Prosecutor can always win this game, since F is unsatisfiable, but we proved that there exist good strategies for Defendant that forces Prosecutor to be able to deal with many different scenarios, i.e., many different partial assignments to $Vars(F)$, which implies lower bounds on resolution refutation length (since a short refutation can be used by Prosecutor to construct a strategy with few records/scenarios).

The key idea in the Defendant strategy for PHP formulas is to pick uniformly at random a matching of $n/4$ pigeons into $n/4$ holes and answer any questions from Prosecutor consistently with this matching. Clearly, there are exponentially many different ways to choose such a matching. What we just showed is that in any complete strategy Prosecutor has to write down a noticeable fraction of (information about) the random matching chosen by Defendant. This implies that Prosecutor has to be able to deal with exponentially many mutually inconsistent partial truth value assignments, which means that the strategy needs to have an exponential number of records.

Our proof used the language of probability theory to show that any particular Prosecutor record has only exponentially small probability of being helpful for Prosecutor for a particular random matching chosen by Defendant. This is just another way of implementing a counting argument saying that Prosecutor needs exponentially many records (since for any random matching there is a helpful record, so the probabilities over all such records has to sum up to 1). It is often the case in proof complexity (and in computational complexity in general) that counting arguments are expressed in terms of probabilities in this way, so it can be good to try to become used to this way of thinking.

References

[Bla37] Archie Blake. *Canonical Expressions in Boolean Algebra*. PhD thesis, University of Chicago, 1937.

[BS97] Roberto J. Bayardo Jr. and Robert Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97)*, pages 203–208, July 1997.

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

[DLL62] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, July 1962.

[DP60] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.

[Hak85] Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39(2-3):297–308, August 1985.

[MMZ+01] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference (DAC '01)*, pages 530–535, June 2001.

[MS99] João P. Marques-Silva and Karem A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, May 1999. Preliminary version in *ICCAD '96*.

[Pud00] Pavel Pudlák. Proofs as games. *American Mathematical Monthly*, pages 541–550, 2000.

[Raz02] Alexander A. Razborov. Proof complexity of pigeonhole principles. In *5th International Conference on Developments in Language Theory, (DLT '01), Revised Papers*, volume 2295 of *Lecture Notes in Computer Science*, pages 100–116. Springer, July 2002.

[Rob65] John Alan Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, January 1965.