DD2242 Seminars on TCS: Proof Complexity Sep 14, 2016

Lecture 4

Lecturer: Jakob Nordström *Scribe: Joseph Swernofsky, Jakob Nordström*

1 Introduction

So far in the course we have investigated the *resolution* proof system. We have obtained lower bounds for pigeonhole principle formulas and Tseitin formulas. Resolution is a very well-studied proof system, and there are several techniques for proving lower bounds, such as:

Prosecutor-Defendant games This is what we have seen already.

Random restrictions This method will feature prominently later in the course for other proof systems.

Width lower bounds For resolution, it turns out that strong enough lower bounds on the width of clauses in a refutation yield length lower bounds. (This was shown in a celebrated paper [BW01], which we will probably only touch on very briefly, if at all, in this course, however.)

In some sense, these approaches are all variations on the same theme, but we do not want to spend time now on elaborating on this and making the connections formal.

Today we will instead start to investigate the *cutting planes* proof system, which is much less well understood. As a case in point, there is essentially only one technique for proving size lower bounds, namely *interpolation*, which works by establishing a connection to *circuit complexity*. In this lecture we will:

- Refresh our memory what circuits are.
- Define and discuss cutting planes.
- Illustrate the interpolation proof technique.

Regarding the final bullet, for starters we will focus on doing interpolation for the resolution proof system (where this technique also works and yields a fundamentally different approach from the ones discussed above). The next two lectures will then extend this to cutting planes.

2 Circuits

Let us start by giving a formal definition of what we mean by a circuit.

Definition 2.1 (Circuit). A *circuit* is a directed acyclic graph (DAG). It has n *sources*, which are nodes labelled by variable inputs, and a unique *sink* without outgoing edges. All non-source vertices are labelled by one of a fixed set of Boolean functions, and are often referred to as *gates*. Typically we require the *fan-in*, which is just another name for the *in-degree* of a vertex, to be at most 2. The standard gates are $\wedge$ (AND) with fan-in 2, $\vee$ (OR) with fan-in 2, and $\neg$ (NOT) with fan-in 1. A circuit C can be thought of as a Boolean function $f_C : \{0,1\}^n \to \{0,1\}$, where the source vertices are the input values, every non-source vertex computes the function it is labelled with on the values provided by its immediate predecessors, and the output of the function is the value computed at the sink. The *size* of a circuit is the total number of vertices in the DAG.

... Potentially insert examples from handwritten lecture notes here at some later stage ...

For $x, y \in \{0,1\}^n$ we write $x \leq y$ if for all $i \in [n]$ it holds that $x_i \leq y_i$. A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is *monotone* if $x \leq y$ implies that $f(x) \leq f(y)$. Remember this key phrase to define

monotone: "Flipping an input bit from 0 to 1 can never flip f from 1 to 0". The example functions xor and sel that we saw on the board in class are *not* monotone, but the majority function maj *is* monotone. A *monotone circuit* is one using AND and OR gates but *no* NOT gates.

Fact 2.2. *A Boolean function $f : \{0,1\}^n \to \{0,1\}^n$ can be computed by a monotone circuit if and only if it is monotone.*

For a family of functions $\{f_n : \{0,1\}^n \to \{0,1\}\}_{n=1}^{\infty}$ we can study the sizes of the smallest circuits computing these functions. This field of research is known as *circuit complexity* and is another line of attack for proving $\mathsf{P} \neq \mathsf{NP}$ (by trying to show something stronger, namely that NP cannot be decided by polynomial-size circuits, or in computational complexity notation that $\mathsf{NP} \subsetneq \mathsf{P/poly}$). Just as proof complexity, this approach has not been terribly successful at reaching its ultimate goal, but there have been many results for restricted subclasses of circuits such as monotone circuits.

3 Cutting Planes

The *cutting planes* proof system, introduced in [CCT87] to formalize the integer linear programming algorithm in [Gom63, Chv73], allows geometric reasoning with linear inequalities with coefficients in $\mathbb{Z}$. As before, we think of "true" as 1 and "false" as 0, and translate a clause

$$C = \bigvee_{x \in P} x \vee \bigvee_{y \in N} \overline{y} \tag{3.1}$$

to the linear inequality

$$\sum_{x \in P} x + \sum_{y \in N} (1 - y) \geq 1 \tag{3.2}$$

which we standardize as

$$\sum_{x \in P} x - \sum_{y \in N} y \geq 1 - |N| \tag{3.3}$$

by bringing all constant terms to the right of the inequality sign. For example, the clause $x \vee y \vee \overline{z}$ becomes $x + y + (1 - z) \geq 1$ or $x + y - z \geq 0$. Note that falsehood (the empty clause) simply translates to $0 \geq 1$, and deriving this inequality will be how we prove the falsity of a CNF formula.

As a side note, it makes perfect sense to consider cutting planes applied to general systems of linear inequalities with integer coefficients, but in this course we will focus on applying cutting planes to CNF formulas translated into inequalities as explained above.

The derivation rules in cutting planes are:

$$\textbf{Variable axioms} \quad \frac{}{x_i \geq 0} \quad \text{and} \quad \frac{}{-x_i \geq -1} \tag{3.4a}$$

$$\textbf{Addition} \quad \frac{\sum_i a_i x_i \geq A \qquad \sum_i b_i x_i \geq B}{\sum_i (a_i + b_i) x_i \geq A + B} \tag{3.4b}$$

$$\textbf{Multiplication} \quad \frac{\sum_i a_i x_i \geq A}{\sum_i c a_i x_i \geq cA} \quad (c \in \mathbb{N}^+) \tag{3.4c}$$

$$\textbf{Division} \quad \frac{\sum_i c a_i x_i \geq A}{\sum_i a_i x_i \geq \lceil A/c \rceil} \quad (c \in \mathbb{N}^+) \tag{3.4d}$$

where a_i, b_i, c, A, and B are all integers and in addition c is positive (as noted above). We want to highlight that in the division rule (3.4d) we can divide with the common factor c on the left and then *divide and round up* the constant term on the right to the closest integer, since we know that we are only interested in $0/1$ solutions. This division rule is where the power of cutting planes lies.

The *length* of a cutting planes refutation is the total number of lines/inequalities in it, and the *size* also sums the sizes of all coefficients (i.e., the bit size of representing them). The natural generalization

of clause space in resolution is to define cutting planes *(line) space* as the maximal number of linear inequalities needed in memory during a refutation, since every clause is translated into a linear inequality, but we will not study space complexity for cutting planes in this course. There is no useful analogue of the width measure for clauses known for cutting planes.

Clearly, cutting planes is sound. It is also implicationally complete, though we will not prove this now. For CNF formulas, which is the case we are interested in, completeness follows from the next lemma.

Lemma 3.1 (Cutting planes efficiently simulates resolution). *If a CNF formula F can be refuted in resolution in length L, then there is a cutting planes refutation in length $\mathrm{O}(L^2)$.*

Proof sketch. Cutting planes can simulate resolution efficiently by mimicking the resolution steps one by one. We leave the details as an exercise. □

This means that cutting planes has the same exponential worst-case upper bound on size as resolution.

Recall that a proof system $\mathcal{P}$ is said to be *exponentially stronger* than another proof system $\mathcal{Q}$ if $\mathcal{P}$ performs at most polynomially worse on every input than $\mathcal{Q}$ but on some family of inputs $\mathcal{Q}$ requires exponentially larger proofs than $\mathcal{P}$.

Theorem 3.2 ([CCT87]). *Cutting planes is exponentially stronger than resolution.*

Proof sketch. By the observation above, cutting planes is never more than polynomially worse than resolution on any input. To see that it is sometimes exponentially better, it suffices to show that the pigeonhole principle formulas have polynomial-size refutations in cutting planes. In cutting planes we can simply count and see that the number of pigeons exceeds the number of holes and from this obtain an immediate contradiction. It is a good exercise to work out the details here to better understand cutting planes. □

To widen our horizons a bit, let us look at another example of formulas exponentially separating resolution and cutting planes (or at least, so it seems), namely the *even colouring (EC) formulas* constructed by Markström [Mar06] and shown in Figure 1. Here one starts with a connected graph $G = (V, E)$ having an Eulerian cycle, i.e., with all vertex degrees even. We identify the edges E with variables $\{x_e \mid e \in E\}$ and write down constraints that edges should be labelled $0/1$ in such a way that for every vertex $v \in V$ the number of 0-edges and 1-edges incident to v is equal. If the total number of edges in the graph is even (as in Figure 1a), then this formula is satisfiable—just fix any Eulerian cycle and label every second edge 0 and 1, respectively. If the number of edges is odd, however, then cutting planes can derive and then sum up the at-least-2 constraints in Figure 1c (the ones with positive coefficients) over all vertices to derive $2 \cdot \sum_{e \in E(G)} x_e \geq |E(G)|$ and then divide by 2 and round up to obtain $\sum_{e \in E(G)} x_e \geq (|E(G)| + 1)/2$. By instead summing up all at-most-2 constraints (the ones with negative coefficients) and dividing by 2 one obtains $\sum_{e \in E(G)} x_e \leq (|E(G)| - 1)/2$, and subtracting these two inequalities yields $0 \geq 1$.

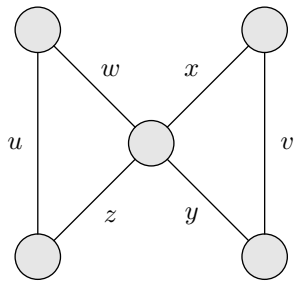
Formalizing the reasoning in the above paragraph yields the following lemma, where the necessary and sufficient condition for unsatisfiability is from [Mar06].

Lemma 3.3. *Let $G = (V, E)$ be an undirected graph with all vertex degrees even. Then the formula $EC(G)$ is unsatisfiable if and only if the number of edges $|E|$ is odd, and cutting planes can refute any unsatisfiable formula $EC(G)$ efficiently.*

For resolution, however, the following seems quite likely to be true.

Claim/Conjecture 3.4. *If $G = (V, E)$ is a "well-connected enough" graph of even degree with $|E|$ odd, then $EC(G)$ is exponentially hard to refute in resolution. (For instance, take G to be a random 6-regular graph on $n = 2m + 1$ vertices, so that the number of edges is odd.)*

This claim-cum-conjecture should be taken with a little grain of salt—it is not formally written down anywhere as far as the lecturer is aware, but should be possible to prove using standard proof complexity machinery as in [BW01] (or at least so the lecturer believes). This provides another example, besides the pigeonhole principles, showing that cutting planes is exponentially stronger than resolution.



(a) Eulerian graph.

$$
\begin{aligned}
&(u \vee w) && \wedge (w \vee x \vee y)\\
\wedge\ &(\overline{u} \vee \overline{w}) && \wedge (w \vee x \vee z)\\
\wedge\ &(u \vee z) && \wedge (w \vee y \vee z)\\
\wedge\ &(\overline{u} \vee \overline{z}) && \wedge (x \vee y \vee z)\\
\wedge\ &(v \vee x) && \wedge (\overline{w} \vee \overline{x} \vee \overline{y})\\
\wedge\ &(\overline{v} \vee \overline{x}) && \wedge (\overline{w} \vee \overline{x} \vee \overline{z})\\
\wedge\ &(v \vee y) && \wedge (\overline{w} \vee \overline{y} \vee \overline{z})\\
\wedge\ &(\overline{v} \vee \overline{y}) && \wedge (\overline{x} \vee \overline{y} \vee \overline{z})
\end{aligned}
$$

(b) Corresponding CNF formula.

$$
\begin{aligned}
u + w &\geq 1\\
-u - w &\geq -1\\
u + z &\geq 1\\
-u - z &\geq -1\\
v + x &\geq 1\\
-v - x &\geq -1\\
v + y &\geq 1\\
-v - y &\geq -1\\
x + y + z + w &\geq 2\\
-x - y - z - w &\geq -2
\end{aligned}
$$

(c) Pseudo-Boolean encoding.

Figure 1: Example of Markström's even colouring (EC) formula (satisfiable instance).

An intriguing fact is that there are SAT solver that use cutting planes reasoning, so-called *pseudo-Boolean solvers*, and instances like pigeonhole principle formulas and even colouring formulas are where these solvers should shine in comparison to CDCL solvers based on resolution.[1] For pigeonhole principle formulas, pseudo-Boolean solvers do indeed perform very well, but although $EC(G)$ is easy for cutting planes, the solvers taking part in the pseudo-Boolean competitions still seem to take exponential time on such formulas. It would be great to understand better why this is the case…

4 Clique-Coclique Formulas and Interpolation

As already mentioned, cutting planes is a poorly understood proof system. We essentially only know one way of proving superpolynomial lower bounds on length for cutting planes, and this is the *interpolation* method introduced by Krajíček [Kra94] and used by Pudlák [Pud97] to establish lower bounds for formulas talking about cliques in and colouring of graphs. We now proceed to give a formal description of these formulas.

The *clique-coclique* formulas are unsatisfiable CNF formulas encoding the contradictory claim that there exist undirected graphs $G = (V, E)$ on $n = |V|$ vertices which have an m-clique but are also $(m-1)$-colourable. The encoding uses the following Boolean variables:

$p_{i,j}$ indicates whether the edge (i, j) is present in G or not (where we enforce $i < j$ since the graph is undirected);

$q_{k,i}$ indicates whether the vertex i in G is the kth member of the m-clique;

$r_{i,\ell}$ indicates whether the vertex i in G has colour ℓ.

The clique-coclique formula consists of the following clauses.

- for each $k \in [m]$, some vertex in G is the kth member of the clique:

$$\bigvee_{i \in [n]} q_{k,i} \ , \tag{4.1a}$$

[1] For this to be possible, though, it is important to make full use of the expressivity of linear inequalities and encode, for instance, the even colouring formulas in so-called pseudo-Boolean form as in Figure 1c rather than as the CNF formula in Figure 1b. Pseudo-Boolean solvers tend not to perform better than CDCL solvers when given CNF input, but explaining why is outside the scope of these notes.

- for all $k, k' \in [m]$, $k \neq k'$, $i \in [n]$, clique vertices have unique member numbers:

$$\overline{q}_{k,i} \vee \overline{q}_{k',i} \ , \tag{4.1b}$$

- for all $k, k' \in [m]$, $k \neq k'$, $i, j \in [n]$, $i < j$, the clique members are connected by edges:

$$p_{i,j} \vee \overline{q}_{k,i} \vee \overline{q}_{k',j} \ , \tag{4.1c}$$

- for each $i \in [n]$, vertex i gets assigned a colour:

$$\bigvee_{\ell \in [m]} r_{i,\ell} \ , \tag{4.1d}$$

- for all $\ell \in [m-1]$, $i, j \in [n]$, $i < j$, neighbouring vertices have distinct colours:

$$\overline{p}_{i,j} \vee \overline{r}_{i,\ell} \vee \overline{r}_{j,\ell} \ . \tag{4.1e}$$

We observe for later use that the clauses in the clique-coclique formula can be split into two parts sharing only the variables $\mathbf{p}$ encoding the edges of the graph. That is, it can be written as $A(\mathbf{p}, \mathbf{q}) \wedge B(\mathbf{p}, \mathbf{r})$ for $A(\mathbf{p}, \mathbf{q})$ being the conjunction of the clauses (4.1a)–(4.1c) and $B(\mathbf{p}, \mathbf{r})$ being the conjunction of the clauses (4.1d)–(4.1e), where the sets of variables $\mathbf{p}, \mathbf{q}, \mathbf{r}$ are disjoint.

Given a partial truth value assignment, or *restriction*, ρ to the variables $Vars(F)$ of a CNF formula F, we write $F\restriction_\rho$ for the new formula obtained by assigning variable values according to ρ and then simplifying F. To simplify, we remove satisfied clauses and falsified literals. For example, for the formula $F = (x \vee y) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z})$ and restriction $\rho = \{z \mapsto 0\} = \{\overline{z}\}$ we obtain $F\restriction_\rho = (x \vee y) \wedge \overline{x}$.

Suppose we have a an unsatisfiable CNF formula in the form $A(\mathbf{p}, \mathbf{q}) \wedge B(\mathbf{p}, \mathbf{r})$ as above (but not necessarily the clique-coclique formula). Notice that when we plug in a particular assignment ρ to $\mathbf{p}$ we get the conjunction of two formulas $A(\mathbf{p}, \mathbf{q})\restriction_\rho = A'(\mathbf{q})$ and $B(\mathbf{p}, \mathbf{r})\restriction_\rho = B'(\mathbf{r})$ on disjoint sets of variables $\mathbf{q}$ and $\mathbf{r}$. Since the original formula was unsatisfiable, at least one of these restricted subformulas must be unsatisfiable.

We say that a Boolean circuit $I(\mathbf{p})$ is an *interpolant* for CNF formula $A(\mathbf{p}, \mathbf{q}) \wedge B(\mathbf{p}, \mathbf{r})$ with disjoint sets of variables $\mathbf{p}, \mathbf{q}, \mathbf{r}$ if for every assignment ρ to the variables $\mathbf{p}$ it holds that $I(\rho) = 0$ implies that $A\restriction_\rho$ is unsatisfiable and $I(\rho) = 1$ implies that $B\restriction_\rho$ is unsatisfiable. In case both subformulas are unsatisfiable the interpolant is free to choose whichever subformula it likes best (but the function has to be well-defined, so it has to make a choice). Note that such an interpolant always exists by definition—we can define a function $I(\mathbf{p})$ that evaluates to 0 whenever $A\restriction_\rho$ is unsatisfiable and takes the value 1 otherwise, and this is a well-defined mathematical function that can be computed by some circuit—but the interpolating circuit might be quite large. We are interested in when the interpolant can be written as a small (polynomial-size) Boolean circuit. It turns out this is possible if $A(\mathbf{p}, \mathbf{q}) \wedge B(\mathbf{p}, \mathbf{r})$ has a short resolution refutation! Flipping this implication around, in the other direction this means that *interpolants can be used to obtain proof complexity lower bounds from circuit complexity lower bounds.*

This suggests the following strategy for proving lower bounds on refutation length:

- Start with a formula $A(\mathbf{p}, \mathbf{q}) \wedge B(\mathbf{p}, \mathbf{r})$.
- Assume, towards contradiction, that the formula has a short resolution refutation.
- Deduce then that there exist a small interpolating circuit.
- Appeal to a(n already known) circuit complexity lower bound saying no such circuit can exist.
- Contradiction! Hence there cannot be a short resolution refutation.

Proof systems for which this strategy works are said to have *feasible interpolation*. Resolution and cutting planes both have feasible interpolation. For resolution this is yet another way to prove lower bounds, but for cutting planes this is the *only* lower bound technique known at the moment. The technique can be used to show that the clique-coclique formulas are hard for cutting planes. Today we will illustrate the proof of the interpolation theorem for resolution and in the next two lectures we will do the lower bound for cutting planes.

5 Statement of Clique-Coclique Formula Lower Bound

In order to establish lower bounds on the refutation length of clique-coclique formulas, we will need the following circuit complexity lower bound.

Theorem 5.1 ([Raz85, AB87]). *Let an undirected graph G be represented by $\binom{n}{2}$ bits encoding its edges and non-edges. Then for $m = \Theta\big(\sqrt[4]{n}\big)$ there is no* monotone *circuit of size $2^{o(\sqrt{m})}$ that can distinguish the following two cases:*

- *G has an m-clique.*
- *G is $(m-1)$-colourable.*

But what an interpolant $I(\mathbf{p})$ for the clique-coclique formula does is exactly to distinguish the two cases in Theorem 5.1. Since an interpolant determines which part of the formula is unsatisfiable for a given assignment to the variables $\mathbf{p}$ encoding the graph, it can separate the cases when G has an m-clique and when it is $(m-1)$-colourable. What Theorem 5.1 says is that every monotone circuit computing such an interpolant must have size $\exp\big(\Omega\big(\sqrt[8]{n}\big)\big)$.

Remark 5.2. The monotonicity assumption in Theorem 5.1 is *very* important. We do not have such strong lower bounds for explicit functions for non-monotone circuits.

In what follows, we will focusing on constructing an interpolant $I(\mathbf{p})$ without caring too much about the (crucial) fact that we want it to be a monotone circuit. Let us define the ternary *selector* function sel by

$$\mathsf{sel}(x,y,z) = \begin{cases} y & \text{if } x = 0, \\ z & \text{if } x = 1. \end{cases} \tag{5.1}$$

We will build circuits with gates $\{\wedge, \vee, \mathsf{sel}\}$. It is not hard to see that sel can be implemented by a subcircuit over $\{\wedge, \vee, \neg\}$ of constant size, so using sel is just a convenient shorthand. A more serious concern is that sel is not a monotone function, but let us decide not to worry about this for now and take care of it at the end of the lecture.

We can now state the theorem that is the main goal of this lecture.

Theorem 5.3 ([Pud97]). *Suppose that $A(\mathbf{p},\mathbf{q}) \wedge B(\mathbf{p},\mathbf{r})$ is an unsatisfiable CNF formula over disjoint sets of variables $\mathbf{p},\mathbf{q},\mathbf{r}$, and that there is a resolution refutation $\pi : A \wedge B \vdash \bot$ in length L. Then the following holds:*

1. *There is an interpolating circuit $I(\mathbf{p})$ over gates $\{\wedge, \vee, \mathsf{sel}\}$ of size $\mathrm{O}(L)$;*
2. *From π we can construct a resolution refutation*

   (a) $\pi_A : A(\rho,\mathbf{q}) \vdash \bot$ if $I(\rho) = 0$, or
   (b) $\pi_B : B(\rho,\mathbf{r}) \vdash \bot$ if $I(\rho) = 1$,

   in both cases of length at most L;
3. *If the $\mathbf{p}$-variables occur only positively in $A(\mathbf{p},\mathbf{q})$ or only negatively in $B(\mathbf{p},\mathbf{r})$, then sel-gates can be replaced by $\wedge$- and $\vee$-gates, yielding a monotone circuit of size $\mathrm{O}(L)$.*

Here is the plan for the proof:

- Fixing $\mathbf{p}$ to ρ, we will split the (restricted) clauses of π into two derivations π_A from $A(\rho,\mathbf{q})$ and π_B from $B(\rho,\mathbf{r})$.
- At least one of π_A and π_B will be assigned the final empty clause and will thus be a resolution refutation of $A(\rho,\mathbf{q})$ or $B(\rho,\mathbf{r})$, respectively.
- We can build a circuit representing our choice of how to split the clauses in π that figures out whether π_A or π_B gets assigned the final clause, and hence which of the formulas $A(\rho,\mathbf{q})$ and $B(\rho,\mathbf{r})$ is unsatisfiable.

6 Proof of Clique-Coclique Formula Lower Bound

Let $\pi = (C_1, \ldots, C_L)$ be any resolution refutation of $A(\mathbf{p}, \mathbf{q}) \wedge B(\mathbf{p}, \mathbf{r})$ and let ρ be any assignment of the $\mathbf{p}$ variables. Let us first prove part 2 of Theorem 5.3.

6.1 Extracting a Resolution Refutation of One of the Subformulas

We start by making a pair of key definitions. For a fixed restriction ρ to $\mathbf{p}$, we define a $\mathbf{q}$-*clause* to be a clause C over variables $\mathbf{q}$ that is in $A(\rho, \mathbf{q})$ or is derivable from $A(\rho, \mathbf{q})$. Similarly, an $\mathbf{r}$-*clause* is a clause C over variables $\mathbf{r}$ which is a member of or derivable from $B(\rho, \mathbf{r})$. As usual, we will let 1 denote the trivial clause that is always satisfied, and we will consider 1 to be derivable from anything so that it qualifies both as a $\mathbf{q}$-clause and as an $\mathbf{r}$-clause.

We will go trough the clauses in $\pi = (C_1, \ldots, C_L)$ in order and construct a sequence of clauses $\widetilde{\pi} = \left(\widetilde{C}_1, \ldots, \widetilde{C}_L\right)$ which will contain the two derivations π_A from $A(\rho, \mathbf{q})$ and π_B from $B(\rho, \mathbf{r})$ mentioned above in our proof plan for Theorem 5.3. More precisely, from each C_i in π we will obtain a clause $\widetilde{C}_i$ satisfying the following properties:

1. $\widetilde{C}_i$ is designated to be either a $\mathbf{q}$-clause or $\mathbf{r}$-clause as defined above, where we write $\mathit{type}\left(\widetilde{C}_i\right) = \mathbf{q}$ or $\mathit{type}\left(\widetilde{C}_i\right) = \mathbf{r}$ to denote the label chosen for $\widetilde{C}_i$.

2. $\widetilde{C}_i = 1$ only if $C_i{\restriction_\rho} = 1$, and if $\widetilde{C}_i \neq 1$ it holds that $\widetilde{C}_i \subseteq C_i \setminus \{a, \overline{a} \mid a \in \rho\}$.

3. With any $\widetilde{C}_i = 1$ we associate an axiom clause E_i that is satisfied by an assignment $\rho(a) = 1$ for some literal $a \in C_i \cap E_i$, where we have $E_i \in A(\mathbf{p}, \mathbf{q})$ if $\mathit{type}\left(\widetilde{C}_i\right) = \mathbf{q}$ and $E_i \in B(\mathbf{p}, \mathbf{r})$ if $\mathit{type}\left(\widetilde{C}_i\right) = \mathbf{r}$.

The clauses $\widetilde{C}_i$ that are mainly of interest to us are nontrivial clauses, but in general we can expect to have a number of trivial clauses since the restriction ρ might satisfy a large portion of the clauses in the formula. For such trivial clauses we can think of the clause E_i in Property 3 as a *justification axiom* with *justification literal* $a \in C_i \cap E_i$ such that $\rho(a) = 1$ explaining why we chose the trivial clause 1 for $\widetilde{C}_i$. These justification axioms and literals will not be needed in our construction of the resolution refutation in part 2 of Theorem 5.3, but will play an important role later when we prove part 3 about monotone circuits in Section 6.3.

We construct $\widetilde{C}_i$ for each $C_i \in \pi$ by forward induction over π. Observe that this is sufficient to establish part 2 of Theorem 5.3. To see this, note that when we reach the final clause $C_L = \bot \in \pi$, by Property 2 we will have that $\widetilde{C}_L \subseteq C_L{\restriction_\rho} = \bot$, i.e., $\widetilde{C}_L = \bot$. Furthermore, $\widetilde{C}_L$ will be labelled as either a $\mathbf{q}$-clause or an $\mathbf{r}$-clause by Property 1, meaning that it is derived from $A(\rho, \mathbf{q})$ only or $B(\rho, \mathbf{r})$ only, respectively. It will be clear from the construction that follows below that the length of this derivation is at most L.

In the base case C_i is an axiom. In this case simply let $\widetilde{C}_i = C_i{\restriction_\rho}$. If C_i is part of $A(\mathbf{p}, \mathbf{q})$, we label $\widetilde{C}_i$ as a $\mathbf{q}$-clause, and if C_i is from $B(\mathbf{p}, \mathbf{r})$, then $\widetilde{C}_i$ is an $\mathbf{r}$-clause. Properties 1 and 2 are clearly satisfied by definition. It is important to note that for many axiom clauses C_i we might have $\widetilde{C}_i = 1$, but that does not violate Property 2. If $\widetilde{C}_i = C_i{\restriction_\rho} = 1$, then we let $E_i = C_i$ be the justification axiom, which obviously fulfils the conditions in Property 3.

For the inductive step, suppose $C_i = C \vee D$ was derived by applying the resolution rule

$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D} \tag{6.1}$$

to two previous clauses $C_j = C \vee x$ and $C_k = D \vee \overline{x}$ for $j, k < i$. By induction, we have already constructed $\widetilde{C}_j$ and $\widetilde{C}_k$ and we know their types as $\mathbf{q}$- or $\mathbf{r}$-clauses. Also, if $\widetilde{C}_j = 1$ and/or $\widetilde{C}_k = 1$, then we have justification axioms E_j and/or E_k that are satisfied by ρ. We make a case analysis over the variable x resolved over depending on whether $x \in \mathbf{p}$, $x \in \mathbf{q}$, or $x \in \mathbf{r}$.

Case 1 ($x \in \mathbf{p}$)**:** If $\rho(x) = 0$, then we set $\widetilde{C}_i = \widetilde{C}_j$ and let $\mathit{type}(\widetilde{C}_i) = \mathit{type}(\widetilde{C}_j)$. Observe that if $C_i\restriction_\rho \neq 1$, then we have $\widetilde{C}_i \subseteq C_j\restriction_\rho \subseteq C_i\restriction_\rho$. If $\widetilde{C}_i = \widetilde{C}_j = 1$, we copy the justification axiom also and let $E_i = E_j$. Note that any justification literal $a \in C_j \cap E_j$ (which cannot be x since $\rho(x) = 0$) is present in C_i as well. If instead $\rho(x) = 1$, then if $C_i\restriction_\rho \neq 1$ we have $\widetilde{C}_k \subseteq C_k\restriction_\rho \subseteq C_i\restriction_\rho$, so we set $\widetilde{C}_i = \widetilde{C}_k$ and let $\widetilde{C}_i$ inherit its type, and possibly its justification axiom E_i, from $\widetilde{C}_k$. This ensures that we maintain Properties 2 and 3. Property 1 holds since if $\rho(x) = 0$, then by the inductive hypothesis we have that $\widetilde{C}_i = \widetilde{C}_j$ is derivable from only $A(\rho, \mathbf{q})$ or only $B(\rho, \mathbf{r})$ depending on its type, and if $\rho(x) = 1$ then the same holds for $\widetilde{C}_i = \widetilde{C}_k$.

As noted above, it might well be that $\widetilde{C}_j = 1$ or $\widetilde{C}_k = 1$ (or both), but we do not care about this. The case analysis based on the value of $\rho(x)$ remains valid, and Properties 1 and 2 are preserved.

Case 2 ($x \in \mathbf{q}$)**:** Here we divide the analysis into subcases depending on the types of $\widetilde{C}_j$ and $\widetilde{C}_k$.

- If exactly one of $\widetilde{C}_j$ or $\widetilde{C}_k$ is an $\mathbf{r}$-clause, set $\widetilde{C}_i$ to that $\mathbf{r}$-clause. If both $\widetilde{C}_j$ or $\widetilde{C}_k$ are $\mathbf{r}$-clauses, arbitrarily pick one of them (say, the one with smallest index). Label $\widetilde{C}_i$ as an $\mathbf{r}$-clause, and if $\widetilde{C}_i = 1$ let it inherit the justification clause from its chosen parent clause. Note that by our inductive hypothesis $\widetilde{C}_i$ constructed in this way will not contain any variable in $\mathbf{q}$. Since $x \in \mathbf{q}$ is the only variable that disappears in the resolution step, this means that we preserve Properties 2 and 3.
- Otherwise, if either $\widetilde{C}_j$ or $\widetilde{C}_k$ is a $\mathbf{q}$-clause not containing x (which is true, for instance, if one if them is a trivial clause 1 labelled by type $\mathbf{q}$), let $\widetilde{C}_i$ equal that $\mathbf{q}$-clause and set $\mathit{type}(\widetilde{C}_i) = \mathbf{q}$. (Again, if both $\widetilde{C}_j$ or $\widetilde{C}_k$ qualify, just arbitrarily pick one of them.) Also, let $\widetilde{C}_i$ inherit the justification clause from its chosen parent if it is trivial.
- Otherwise, if either $\widetilde{C}_j$ or $\widetilde{C}_k$ is a $\mathbf{q}$-clause not containing $\overline{x}$, then let $\widetilde{C}_i$ equal that $\mathbf{q}$-clause (or choose a clause arbitrarily if both qualify), label $\widetilde{C}_i$ a $\mathbf{q}$-clause, and let it inherit the justification clause from its parent if needed.
- If none of the above cases apply, then we have two $\mathbf{q}$-clauses $\widetilde{C}_j = \widetilde{C}'_j \vee x$ and $\widetilde{C}_k = \widetilde{C}'_k \vee \overline{x}$ which are both nontrivial (i.e., distinct from 1). Set $\widetilde{C}_i$ to be the resolvent $\widetilde{C}'_j \vee \widetilde{C}'_k$ of these two clauses and let $\mathit{type}(\widetilde{C}_i) = \mathbf{q}$. (Note that this is the first time we actually used the resolution rule to construct $\widetilde{C}_i$.)

By construction, $\widetilde{C}_i$ will not contain x and it can be verified that we will only have $\widetilde{C}_i = 1$ if $C_i\restriction_\rho = 1$ (using again that $x \in \mathbf{q}$ is the only variable that disappears in the resolution step, and recalling that ρ does not assign values to any variables in $\mathbf{q}$). Hence, Property 2 holds. Property 3 holds since no variables in $\mathbf{p}$ disappear in the resolution step. For Property 1, just observe that if $\widetilde{C}_i$ gets classified as an $\mathbf{r}$-clause then no resolution step is involved, and if the end result is a $\mathbf{q}$-clause obtained by resolution, then both premises are $\mathbf{q}$-clauses derivable from $A(\rho, \mathbf{q})$ and so this holds also for their resolvent.

Case 3 ($x \in \mathbf{r}$)**:** this case is analogous to the case $x \in \mathbf{q}$ but exchanging the roles of $\mathbf{r}$- and $\mathbf{q}$-variables. We leave the details to the reader.

As explained above, part 2 of Theorem 5.3 now follows by the induction principle.

6.2 Writing down the Interpolating Circuit

We proceed to prove part 1 of Theorem 5.3. We will use the construction in Section 6.1 with the labelling of the clauses $\widetilde{C}_i$ as $\mathbf{q}$- or $\mathbf{r}$-clauses to build the desired interpolant $I(\mathbf{p})$. Note that at the very end of the process we label the final clause $\widetilde{C}_L$ as either a $\mathbf{q}$-clause or an $\mathbf{r}$-clause, and this tells us whether $A(\rho, \mathbf{q})$ or $B(\rho, \mathbf{r})$ is unsatisfiable. All that we need to do is to build a circuit that performs the same kind of classification of the clauses in the refutation until we know what type is assigned to $\widetilde{C}_L$.

We will build this circuit $I(\mathbf{p})$ using gates $\{\vee, \wedge, \mathsf{sel}\}$ and constants $\{0, 1\}$.[2] As we construct the circuit we will associate the vertices v_i in it with the clauses $C_i \in \pi$. We will maintain the invariant that if $\widetilde{C}_i$ is labelled as a $\mathbf{q}$-clause under some assignment ρ of $\mathbf{p}$, then the value computed at v_i in the circuit on input ρ is 0, and if $\widetilde{C}_i$ is labelled as an $\mathbf{r}$-clause, then v_i computes value 1. The output of the circuit, which is the type of $\widetilde{C}_L = \bot$, will then tell us that $A(\rho, \mathbf{q})$ is unsatisfiable if $I(\rho) = 0$ and that $B(\rho, \mathbf{r})$ is unsatisfiable if $I(\rho) = 1$. This is all that we need to show part 1 of the theorem.

More formally, as the blueprint for our circuit $I(\mathbf{p})$ we will take the DAG representation G_π of the resolution refutation π. For every clause $C_i \in \pi$ we will label the vertex v_i in $I(\mathbf{p})$ with a suitable gate taking suitable inputs. As in Section 6.1, we argue by forward induction over $\pi = (C_1, \ldots, C_L)$.

If C_i is an axiom in $A(\mathbf{p}, \mathbf{q})$ we fix v_i to the constant 0, otherwise if it belongs to $B(\mathbf{p}, \mathbf{r})$ we fix v_i to 1. Remember that we want to maintain the invariant that $\mathbf{q}$-clauses correspond to vertices v_i computing 0 and $\mathbf{r}$-clauses correspond to vertices v_i computing 1. So far, so good.

If C_i was derived by resolution from $C_j = C \vee x$ and $C_k = D \vee \overline{x}$ for $j, k < i$, we have the same kind of case analysis as in Section 6.1. Let us overload $\mathit{type}\big(\widetilde{C}_i\big)$ to denote the *binary value* of the type of the clause as defined above, so that

$$\mathit{type}\big(\widetilde{C}_i\big) = \begin{cases} 0 & \text{if } \widetilde{C}_i \text{ is a } \mathbf{q}\text{-clause,} \\ 1 & \text{if } \widetilde{C}_i \text{ is a } \mathbf{r}\text{-clause.} \end{cases} \tag{6.2}$$

We now choose the gate for v_i as follows.

Case 1 ($x \in \mathbf{p}$)**:** Mark vertex v_i with the selector function sel taking as inputs x and the outputs of v_j and v_k (which by our inductive hypothesis compute $\mathit{type}\big(\widetilde{C}_j\big)$ and $\mathit{type}\big(\widetilde{C}_k\big)$, respectively). It is straightforward to verify that the way the type of $\widetilde{C}_i$ is determined in Section 6.1 is by computing

$$\mathit{type}\big(\widetilde{C}_i\big) = \mathsf{sel}\big(x, \mathit{type}\big(\widetilde{C}_j\big), \mathit{type}\big(\widetilde{C}_k\big)\big) = \mathsf{sel}(x, v_j, v_k) \ . \tag{6.3}$$

Case 2 ($x \in \mathbf{q}$)**:** Mark vertex v_i with an OR-gate $\vee$ taking v_j and v_k as inputs. If at least one of $\widetilde{C}_j$ or $\widetilde{C}_k$ has been classified as an $\mathbf{r}$-clause, which by our inductive hypothesis means that either v_j or v_k computes the value 1, then $\widetilde{C}_i$ gets classified as an $\mathbf{r}$-clause, and otherwise it becomes a $\mathbf{q}$-clause. This is just another way of saying that $\mathit{type}\big(\widetilde{C}_i\big) = \mathit{type}\big(\widetilde{C}_j\big) \vee \mathit{type}\big(\widetilde{C}_k\big)$, which is exactly the values that v_i now computes.

Case 3 ($x \in \mathbf{r}$)**:** Mark vertex v_i with an AND-gate $\wedge$ taking inputs v_j and v_k. This is (anti-)symmetric to the case when $x \in \mathbf{q}$, and as in Section 6.1 we leave the details of this case to the reader.

6.3 Removing Selector Gates

We have constructed an interpolating circuit and have proven parts 1 and 2 of our main theorem for today. To establish part 3 we need to prove that if the $\mathbf{p}$-variables occur only positively in $A(\mathbf{p}, \mathbf{q})$ or only negatively in $B(\mathbf{p}, \mathbf{r})$, then we can change the circuit above slightly by replacing the sel-gates with small monotone subcircuits. This will give us a monotone interpolating circuit.

… Potentially insert a picture with the sel function here at some later stage …

Let as assume that the $\mathbf{p}$-variables appear only positively in $A(\mathbf{p}, \mathbf{q})$. In this case we choose to replace all occurrences of

$$\mathsf{sel}(x, a, b) = (x \vee a) \wedge (\overline{x} \vee b) \tag{6.4}$$

by the function

$$(x \vee a) \wedge b \ . \tag{6.5}$$

[2] We did not have constants in Definition 2.1, but if this troubles us we can get rid of them in a postprocessing step by propagating simplifications $0 \wedge x = 0$, $1 \wedge x = x$, $0 \vee x = x$, and $1 \vee x = 1$ through the circuit until all constants have been removed.

What this means is that we replace sel-gates in (6.3), the only case where a non-monotone gate could be introduced in $I(\mathbf{p})$, with the computation

$$type\big(\widetilde{C}_i\big) = \big(x \vee type\big(\widetilde{C}_j\big)\big) \wedge type\big(\widetilde{C}_k\big) \quad . \tag{6.6}$$

A closer study of the functions in (6.4) and (6.5) reveals that the only differ in that the monotone function in (6.5) returns 0 instead of 1 on input $(x, a, b) = (0, 1, 0)$.

When does this happen in our proof? This is when we are in the first case in our case analysis, i.e., when we have $C_i = C \vee D$ derived from $C_j = C \vee x$ and $C_k = D \vee \overline{x}$ for $x \in \mathbf{p}$ such that $\rho(x) = 0$. Moreover, $\widetilde{C}_j$ has been classified as an $\mathbf{r}$-clause and $\widetilde{C}_k$ as a $\mathbf{q}$-clause under the current restriction ρ. According to our original case analysis we should have set $\widetilde{C}_i = \widetilde{C}_j$ and classified $\widetilde{C}_i$ as an $\mathbf{r}$-clause. Instead, according to (6.5) we set $\widetilde{C}_i = \widetilde{C}_k$, which is a $\mathbf{q}$-clause.

Why is this a problem? The clause $\widetilde{C}_k$ does not contain $x \in \mathbf{p}$ by construction, and so if $\widetilde{C}_k$ is nontrivial Property 2 holds since $x \in \mathbf{p}$ is the only variable that disappears in the resolution step. It is also easy to see that Property 1 always holds by the inductive hypothesis when we copy a clause. What is more problematic, though, is if we have a resolvent C_i such that

$$C_i\restriction_\rho = (C \vee D)\restriction_\rho \neq 1 \tag{6.7}$$

but have chosen $\widetilde{C}_k = 1$ since

$$C_k\restriction_\rho = (D \vee \overline{x})\restriction_\rho = 1 \quad . \tag{6.8}$$

If this is the case, then setting $\widetilde{C}_i = \widetilde{C}_k$ violates Property 2 in the definition of $\mathbf{q}$- and $\mathbf{r}$-clauses, since now $\widetilde{C}_i = 1$ but $C_i\restriction_\rho \neq 1$. Should this happen, the whole proof crashes and burns. Not good.

Let us analyse this worrying scenario more closely. If $\widetilde{C}_k = 1$, then by Property 3 in our construction there is also a justification axiom clause E_k satisfied by an assignment $\rho(a) = 1$ for some literal $a \in C_k \cap E_k$, where in addition $E_k \in A(\mathbf{p}, \mathbf{q})$ since $\widetilde{C}_k$ is a $\mathbf{q}$-clause. In view of (6.7) and (6.8), the justification literal a must satisfy $a \in (D \vee \overline{x}) \setminus (C \vee D)$, i.e., we must have $a = \overline{x}$ satisfied by the assignment $\rho(x) = 0$. But this means that the axiom clause $E_k \in A(\mathbf{p}, \mathbf{q})$ would need to contain the literal $\overline{x}$, and by assumption variables $x \in \mathbf{p}$ appear only positively in $A(\mathbf{p}, \mathbf{q})$. So we do not need to worry—this problematic scenario never arises. Although the computation for v_i in the monotone version of the circuit seems to make a mistake for the input $(x, a, b) = (0, 1, 0)$, the previously constructed clause $\widetilde{C}_k$ is guaranteed to be nice enough for the invariants to be preserved when we set $\widetilde{C}_i = \widetilde{C}_k$ and $type\big(\widetilde{C}_i\big) = type\big(\widetilde{C}_k\big)$.

An analogous hack works if $\mathbf{p}$-variables instead appear only negatively in $B(\mathbf{p}, \mathbf{r})$. We leave the details to the reader. This concludes our proof of part 3 of Theorem 5.3.

7 Summing Up and Looking Forward

We have now established Theorem 5.3. If we apply this theorem to the formulas in (4.1a)–(4.1e) and combine it with Theorem 5.1, then we can deduce that for the clique-coclique formulas for graphs over n vertices with $m = \Theta\big(\sqrt[4]{n}\big)$ it holds that resolution needs refutations of length $\exp\big(\Omega\big(n^\delta\big)\big)$ for $\delta = 1/8$.

In the next two lectures we will prove the same lower bound for cutting planes. To do so will require us to lift the interpolation technique from Boolean circuits to *real* circuits, which are circuits computing with arbitrary real numbers instead of just Boolean values $\{0, 1\}$. But enough for today!

References

[AB87] Noga Alon and Ravi B. Boppana. The monotone circuit complexity of Boolean functions. *Combinatorica*, 7(1):1–22, March 1987.

[BW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *Journal of the ACM*, 48(2):149–169, March 2001. Preliminary version in *STOC '99*.

[CCT87] William Cook, Collette Rene Coullard, and György Turán. On the complexity of cutting-plane proofs. *Discrete Applied Mathematics*, 18(1):25–38, November 1987.

[Chv73] Vašek Chvátal. Edmond polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4(1):305–337, 1973.

[Gom63] Ralph E. Gomory. An algorithm for integer solutions of linear programs. In R.L. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*, pages 269–302. McGraw-Hill, New York, 1963.

[Kra94] Jan Krajíček. Lower bounds to the size of constant-depth propositional proofs. *Journal of Symbolic Logic*, 59(1):73–86, 1994.

[Mar06] Klas Markström. Locality and hard SAT-instances. *Journal on Satisfiability, Boolean Modeling and Computation*, 2(1-4):221–227, 2006.

[Pud97] Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *Journal of Symbolic Logic*, 62(3):981–998, September 1997.

[Raz85] Alexander A. Razborov. Lower bounds for the monotone complexity of some Boolean functions. *Soviet Mathematics Doklady*, 31(2):354–357, 1985. English translation of a paper in *Doklady Akademii Nauk SSSR*.