Proof Complexity as a Computational Lens 2025/26 Jan 20, 2026

Lecture 20

Lecturer: Jakob Nordström Scribe: Susanna de Rezende, Joseph Swernofsky, Name-of-scribe-here

1 Introduction

So far in the course we have investigated the proof systems *resolution* and *polynomial calculus*.

Resolution is a very well-studied proof system, and there are several techniques for proving lower bounds. We mainly focused on

- Prosecutor-Defendant games,
- random restrictions, and
- size-width relations.

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. For polynomial calculus, the menu of options was much more limited. Although we used random restrictions to establish that low-degree proofs could have maximal possible size, almost all the other size lower bounds we did for this proof system boiled down to showing degree lower bounds and then using the size-degree relation.

Today we will start to study the *cutting planes* proof system, which is much less well understood. As a case in point, for a long time there has essentially been only one technique for proving size lower bounds, namely *interpolation*, which works by establishing a connection to *circuit complexity*. We acquainted ourselves with interpolation early on in the course for resolution, but this was really intended as a warm-up for what is about to follow now. In this lecture we will:

- Provide a formal definition of the cutting planes proof system and review some preliminaries.
- Refresh our memory regarding what interpolation is.
- Use interpolation to prove size lower bounds for cutting planes (except that as for resolution we will not do the details of the circuit lower bound needed to obtain the final result).

In coming lectures, we will discuss a few more recent proof techniques that have been developed for the cutting planes proof system.

2 Cutting Planes Definition and Preliminaries

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

$$C = \bigvee_{x \in P} x \vee \bigvee_{y \in N} \overline{y} \tag{2.1}$$

to the linear inequality

$$\sum_{x \in P} x + \sum_{y \in N} (1 - y) \geq 1 \tag{2.2}$$

which we standardize as

$$\sum_{x \in P} x - \sum_{y \in N} y \geq 1 - |N| \tag{2.3}$$

by bringing all constant terms to the right of the inequality sign. For example, the clause $x \lor y \lor \overline{z}$ 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 we will mostly focus on applying cutting planes to CNF formulas translated into inequalities as explained above.

The derivation rules in cutting planes as it is usually derived in the proof complexity literature are

$$\textbf{Variable axioms} \quad \frac{}{x_i \geq 0} \quad \text{and} \quad \frac{}{-x_i \geq -1} \tag{2.4a}$$

$$\textbf{Addition} \quad \frac{\sum_i a_i x_i \geq A \qquad \sum_i b_i x_i \geq B}{\sum_i (a_i + b_i) x_i \geq A + B} \tag{2.4b}$$

$$\textbf{Multiplication} \quad \frac{\sum_i a_i x_i \geq A}{\sum_i c a_i x_i \geq cA} \quad (c \in \mathbb{N}^+) \tag{2.4c}$$

$$\textbf{Division} \quad \frac{\sum_i c a_i x_i \geq A}{\sum_i a_i x_i \geq \lceil A/c \rceil} \quad (c \in \mathbb{N}^+) \tag{2.4d}$$

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 (2.4d) 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-valued solutions. This division rule is where the power of cutting planes lies.

The division rule is modelling what is referred to as *Chvátal-Gomory cuts* in the operations research literature. An alternative definition of division, which is closer to what is used in practice, is

$$\textbf{General division} \quad \frac{\sum_i a_i x_i \geq A}{\sum_i \lceil a_i/c \rceil x_i \geq \lceil A/c \rceil} \quad (c \in \mathbb{N}^+) \, . \tag{2.5}$$

To see that this rule is valid, note that $\sum_i a_i x_i \geq A$ clearly implies that $\sum_i \frac{a_i}{c} x_i \geq \frac{A}{c}$, and this is only strenghted by rounding up on the left-hand side to get $\sum_i \lceil \frac{a_i}{c} \rceil x_i \geq \frac{A}{c}$. Since the left-hand side is integral, it must now hold that $\sum_i \lceil \frac{a_i}{c} \rceil x_i \geq \lceil \frac{A}{c} \rceil$. We will think of cutting planes as being defined with this more general division rule when convenient.

Sometimes it is also helpful to write the 0–1 integer linear inequalities in so-called *normalized form* as a positive linear combination of literals $\sum_i a_i \ell_i \geq A$ for $a_i, A \in \mathbb{N}^+$, where the constant term A is referred to as the *degree of falsity* (or just *degree*). In this setting, the clause $C = \bigvee_{x \in P} x \lor \bigvee_{y \in N} \overline{y}$ is simply translated to

$$\sum_{x \in P} x + \sum_{y \in N} \overline{y} \geq 1 \tag{2.6}$$

saying that the sum of the literal in the clause is at least one. Normalized form is particularly useful when cutting planes is used as the reasoning method in so-called *pseudo-Boolean solvers*, some of which use the rule

$$\textbf{Saturation} \quad \frac{\sum_i a_i \ell_i \geq A}{\sum_i \min(a_i, A) \cdot \ell_i \geq A} \tag{2.7}$$

instead of division. The saturation rule says that no coefficient on the right need to be larger than the degree of falsity. Note that this rule is only sound if coefficients are non-negative. Since we will focus on theoretical aspects of the cutting planes proof system in our lectures, we will represent our 0–1 linear inequalities in terms of linear combinations of variables unless stated otherwise.

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. 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*—i.e., if a set of 0–1 linear inequalities imply some other 0–1 linear inequality, then there is also a syntactic cutting planes derivation of this fact—though we will not prove this now. For CNF formulas, which is the case we are interested in, the fact that cutting planes is *refutationally complete*—i.e., can derive contradiction from any unsatisfiable formula—follows from the next lemma.

Lemma 2.1 (Cutting planes efficiently simulates resolution). *If a CNF formula F can be refuted in resolution in length L and clause space s, then there is a cutting planes refutation (with general division) in length $\mathrm{O}(L)$ and line space $\mathrm{O}(s)$.*

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 2.2 ([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. □

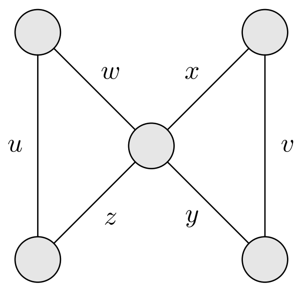
To widen our horizons a bit, let us look at another example of formulas exponentially separating resolution and cutting planes, namely the *even colouring (EC) formulas* constructed by Markström [Mar06] and shown in Figure 1. These formulas are defined in terms of connected graphs $G = (V, E)$ having 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 $0 \geq 1$. It is not hard to show that the 0–1 linear encoding in in Figure 1c can be derived from Figure 1b.

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

Lemma 2.3. *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 tree-like cutting planes can refute any unsatisfiable formula $EC(G)$ efficiently.*

For resolution, however, the even colouring formulas are hard for the right type of graphs G.

Theorem 2.4 ([AR22]). *If $G = (V, E)$ is a "well-connected enough" graph of large enough even degree with $|E|$ odd, then $EC(G)$ is exponentially hard to refute in resolution (and also in polynomial calculus over field of characteristic distinct from 2).*



(a) Eulerian graph.

$$
\begin{array}{rl}
(u \lor w) & \land (w \lor x \lor y) \\
\land (\overline{u} \lor \overline{w}) & \land (w \lor x \lor z) \\
\land (u \lor z) & \land (w \lor y \lor z) \\
\land (\overline{u} \lor \overline{z}) & \land (x \lor y \lor z) \\
\land (v \lor x) & \land (\overline{w} \lor \overline{x} \lor \overline{y}) \\
\land (\overline{v} \lor \overline{x}) & \land (\overline{w} \lor \overline{x} \lor \overline{z}) \\
\land (v \lor y) & \land (\overline{w} \lor \overline{y} \lor \overline{z}) \\
\land (\overline{v} \lor \overline{y}) & \land (\overline{x} \lor \overline{y} \lor \overline{z})
\end{array}
$$

(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 (but for a satisfiable instance).

The bound obtained in [AR22] requires graph of very large (but constant) degree, and the lower bound is $\exp(\Omega(n/\log n))$ instead of $\exp(\Omega(n))$. It seems like it should be possible to prove truly exponential lower bounds by taking G to be a random 6-regular or 10-regular graph on $n = 2m + 1$ vertices, so that the number of edges is odd, and then use fairly standard proof complexity machinery as in [BW01], but the lecturer is not aware of that anyone would have worked through the calculations (which might be a good exercise) In any case, even colouring formulas provide another example, besides the pigeonhole principles, that cutting planes is exponentially stronger than resolution.

3 Brief Interlude on Pseudo-Boolean Solving

THE TEXT BELOW IS BADLY OUT OF DATE AND WOULD BE GOOD TO UPDATE

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…

THE TEXT ABOVE IS BADLY OUT OF DATE AND WOULD BE GOOD TO UPDATE

4 Recap of Clique-Colouring Formulas and Interpolation

As already mentioned, cutting planes is a poorly understood proof system. The first, and for a long time only, method for proving superpolynomial length on the lower bounds on length for cutting planes 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.

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

[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 this lecture.

- $p_{i,j}$, for $i, j \in [n]$, $i < j$, representing the truth value of “there is an edge between vertices i and j;”
- $q_{k,i}$, for $i \in [n]$ and $k \in [m]$ representing the truth value of “vertex i is the kth member of the m-clique;” and
- $r_{i,\ell}$, for $i \in [n]$ and $\ell \in [m-1]$ representing the truth value of “vertex i is has colour ℓ.”

The clique-colouring formula consists of the following clauses (where we write $(x_1 \wedge \cdots \wedge x_m) \rightarrow y$ to denote the equivalent clause $\overline{x}_1 \vee \cdots \vee \overline{x}_m \vee y$ to highlight that a clause on this form can be viewed as encoding an implication):

$$
\begin{aligned}
& q_{k,1} \vee q_{k,2} \vee \cdots \vee q_{k,n} && k \in [m] && \text{(4.1a)} \\
& \overline{q}_{k,i} \vee \overline{q}_{k',i} && i \in [n];\ k, k' \in [m], k \neq k' && \text{(4.1b)} \\
& (q_{k,i} \wedge q_{k',j}) \rightarrow p_{i,j} && i, j \in [n], i < j;\ k, k' \in [m], k \neq k' && \text{(4.1c)} \\
& r_{i,1} \vee r_{i,2} \vee \cdots \vee r_{i,m-1} && i \in [n] && \text{(4.1d)} \\
& (r_{i,\ell} \wedge r_{j,\ell}) \rightarrow \overline{p}_{i,j} && i, j \in [n], i < j;\ \ell \in [m-1] && \text{(4.1e)}
\end{aligned}
$$

where (4.1a)–(4.1b) encodes that the graph has an m-clique and (4.1d)–(4.1e) that it is $(m-1)$-colourable.

We observe for later use that the clauses in the clique-colouring 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.

Recall that given a partial truth value assignment, or *restriction*, ρ to the variables $\mathit{Vars}(F)$ 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-colouring 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 the formula $A(\mathbf{p}, \mathbf{q}) \wedge B(\mathbf{p}, \mathbf{r})$ has a short 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 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 no such short refutation can exist.

Proof systems for which this strategy works are said to have *feasible interpolation*. For the final step to work and yield unconditional lower bounds it is important that the interpolating circuit is *monotone*, because it is currently not known how to prove strong enough lower bounds for general, non-monotone, circuits. This is referred to as *monotone feasible interpolation*. We saw in an earlier lecture that resolution has monotone feasible interpolation, and today we will prove an analogous result for cutting planes.

To obtain the lower bounds for resolution, we used the following result in circuit complexity.

Theorem 4.1 ([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.*

We observed that what an interpolant $I(\mathbf{p})$ for the clique-colouring formula does is exactly to distinguish the two cases in Theorem 4.1. 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 4.1 says is that every monotone circuit computing such an interpolant must have size $\exp\big(\Omega\big(\sqrt[8]{n}\big)\big)$.

To obtain lower bounds for cutting planes, we use the same overall proof strategy, but in order to extract interpolating circuits from cutting planes refutations we have to use a more general class of *real circuits*, which are circuits that compute with real numbers and have arbitrary real functions as gates. Our goal for today is to show how to build small interpolants in the form of monotone real circuits from short cutting planes refutations of clique-colouring formulas. After this, we will sketch the main ideas in the proof that these formulas cannot have such small interpolating circuits, but due to time constraints we will not do the full proof.

As already explained, when we want to refute a CNF formula in cutting planes we translate clauses to inequalities by replacing connectives $\lor$ with addition and negated variables $\overline{x}$ with $(1-x)$. When converting the clauses (4.1b) in clique-colouring formulas to inequalities, however, we can note that a set of clauses

$$\left\{\overline{x}_i \lor \overline{x}_j \,\middle|\, 1 \le i < j \le s\right\}, \tag{4.2a}$$

which is translated to

$$\left\{-x_i - x_j \ge -1 \,\middle|\, 1 \le i < j \le s\right\}, \tag{4.2b}$$

is just another way to encode the constraint

$$\sum_{i=1}^{s} x_i \le 1\,, \tag{4.2c}$$

and since (4.2c) can be written natively as just one linear constraint

$$\sum_{i=1}^{s} -x_i \ge -1 \tag{4.2d}$$

in the context of cutting planes (where we switched to negations just to keep the format $\sum_i c_i x_i \ge K$ of greater-than inequalities that we have adopted for cutting planes) it seems more natural to use the latter encoding rather than a quadratic number of constraints as in (4.2b). For our purposes, it does not really matter which form we use since one can derive (4.2b) from (4.2d) and vice-versa in polynomial length. In one direction (the one we care about in order to get lower bounds for refutations of CNF formulas) this is obvious—from $\sum_{i=1}^{s} -x_i \ge -1$ one can derive $-x_{i_1} - x_{i_2} \ge -1$ for any $i_1, i_2 \in [s]$, $i_1 \ne i_2$, by just adding axioms $x_i \ge 0$ for all $i \in [s] \setminus \{i_1, i_2\}$; the other direction is also straightforward. This means that superpolynomial length lower bounds for clique-colouring formulas using the encoding in (4.2d) imply superpolynomial bounds for the encoding using (4.2b) and the other way round. In view of this, we will define the clique-colouring formula in cutting planes as consisting of the inequalities

$$q_{k,1} + q_{k,2} + \cdots + q_{k,n} \geq 1 \qquad k \in [m] \tag{4.3a}$$
$$-q_{1,i} - q_{2,i} - \cdots - q_{m,i} \geq -1 \qquad i \in [n] \tag{4.3b}$$
$$p_{i,j} - q_{k,i} - q_{k',j} \geq -1 \qquad i, j \in [n], i < j;\ k, k' \in [m], k \neq k' \tag{4.3c}$$
$$r_{i,1} + r_{i,2} + \cdots + r_{i,m-1} \geq 1 \qquad i \in [n] \tag{4.3d}$$
$$-p_{i,j} - r_{i,\ell} - r_{j,\ell} \geq -2 \qquad i, j \in [n], i < j;\ \ell \in [m-1] \tag{4.3e}$$

and prove a lower bound on the length of cutting planes derivations establishing that this set of linear constraints is inconsistent.

5 Interpolation for Cutting Planes

Since lines in a cutting planes proofs consist of linear inequalities and the coefficients can be arbitrary integers, we need a more general computational model than standard Boolean circuits when we want to construct interpolants from cutting planes proofs.

Definition 5.1 (Real circuit [Pud97]). A *real circuit* C is a directed acyclic graph (DAG) with s sources, which are vertices labelled by variable inputs, and a unique sink. Every non-source vertex v with fan-in d_v is labelled by a function $f_v : \mathbb{R}^{d_v} \to \mathbb{R}$ of arity matching the fan-in of the vertex, and we will also refer to such vertices v as *gates*. The incoming edges from the predecessors of v are ordered in some fixed way, so that the function f_v labelling v is well-defined, and every gate v computes the value of the function f_v applied to the values provided by its predecessors. We require all vertices to have bounded fan-in $\mathrm{O}(1)$, and although the exact bound is not too important in what follows we will insist on fan-in at most 2. The output of the circuit C is the value computed by the sink vertex, and this defines a function $f_C : \mathbb{R}^s \to \mathbb{R}$ on the inputs in the natural way, which is the function computed by the circuit. The *size* of a circuit is the total number of vertices in the DAG.

A real circuit is *monotone* if all gates in it compute non-decreasing functions. We say that a real circuit computes a Boolean function if when restricted to inputs in $\{0,1\}^s$ it produces an output in $\{0,1\}$ (but in intermediate steps the circuit can still work with arbitrary real numbers).

Gates in a monotone real circuit can emulate Boolean gates such as $\wedge$ and $\vee$ with for example $\max$ and $\min$, respectively, so monotone real circuits are at least as strong as monotone Boolean circuits. Are they stronger, and if so how much? Well, all we know is that there exist functions computable by linear-size real monotone circuits that require exponential-size monotone Boolean circuits, but we do not know of any explicit example of such a function [Ros97].

We can now formally state the theorem we will prove today.

Theorem 5.2 ([Pud97]). *Let π be a cutting planes refutation of the inequalities*

$$\sum_{p \in \mathbf{p}} c_{i,p} p + \sum_{q \in \mathbf{q}} c_{i,q} q \geq D_i \qquad i \in I \tag{5.1a}$$
$$\sum_{p \in \mathbf{p}} c_{j,p} p + \sum_{r \in \mathbf{r}} c_{j,r} r \geq D_j \qquad j \in J \tag{5.1b}$$

for disjoint sets of variables $\mathbf{p}$, $\mathbf{q}$, and $\mathbf{r}$, and let w be the maximal number of $\mathbf{p}$-variables that appear in any of these inequalities. Then if all coefficients $c_{i,p}$ for $i \in I$ are non-negative or if all coefficients $c_{j,p}$ for $j \in J$ are non-positive, it holds that the formula has an interpolant in the form of a monotone real circuit of size at most $w \cdot L(\pi)$.

We will refer to inequalities (5.1a) as $\mathbf{q}$-axioms and inequalities (5.1b) as $\mathbf{r}$-axioms.

As defined, the clique-colouring formula already gives us a natural partition $\mathbf{p}$, $\mathbf{q}$, and $\mathbf{r}$ of the variables. Moreover, it is easy to see that if we partition the inequalities of the clique-colouring formula so that (4.3a)–(4.3c) correspond to $\mathbf{q}$-axioms and (4.3d)–(4.3e) correspond to $\mathbf{r}$-axioms, then the conditions of the theorem are satisfied. Since there is at most one $\mathbf{p}$-variable in each inequality, we immediately get the following corollary.

Corollary 5.3 ([Pud97]). *Let π is a cutting planes proof refutation of the clique-colouring formula. Then the formula has an interpolant in the form of a monotone real circuit of size at most $L(\pi)$.*

6 Proof of Cutting Planes Interpolation Theorem

We proceed to prove Theorem 5.2. Let π be a cutting planes refutation of a set of inequalities as in the statement of the theorem and let ρ be an assignment to the $\mathbf{p}$-variables.

Our plan is to split the refutation $\pi\restriction_\rho$ in to two derivations, one from restricted $\mathbf{q}$-axioms and one from restricted $\mathbf{r}$-axioms, in such a way that at least one of these will be deriving contradiction. We will then build an interpolating real circuit for the formula using one of these derivations and finally argue that this can be done by a monotone circuit.

Consider the following generic cutting planes line

$$\sum_{p\in\mathbf{p}} c_p p + \sum_{q\in\mathbf{q}} c_q q + \sum_{r\in\mathbf{r}} c_r r \geq D \tag{6.1}$$

representing an inequality in the refutation π. When restricted by the assignment ρ, this inequality becomes

$$\sum_{q\in\mathbf{q}} c_q q + \sum_{r\in\mathbf{r}} c_r r \geq D - \sum_{p\in\mathbf{p}} c_p \rho(p)\,. \tag{6.2}$$

In order to build two derivations, one from $\mathbf{q}$-axioms and one from $\mathbf{r}$-axioms, we will replace each such line (6.2) in $\pi\restriction_\rho$ with two inequalities

$$\sum_{q\in\mathbf{q}} c_q q \geq D_{\mathbf{q}} \tag{6.3a}$$

and

$$\sum_{r\in\mathbf{r}} c_r r \geq D_{\mathbf{r}}\,, \tag{6.3b}$$

where $D_{\mathbf{q}}$ and $D_{\mathbf{r}}$ are integers to be chosen appropriately as described next. Note that the sum of the left-hand side of (6.3a) and the left-hand side of (6.3b) equals the left-hand side of (6.2). For the right-hand sides, the choices of $D_{\mathbf{q}}$ and $D_{\mathbf{r}}$ will be such that the inequality

$$D_{\mathbf{q}} + D_{\mathbf{r}} \geq D - \sum_{p\in\mathbf{p}} c_p \rho(p) \tag{6.4}$$

holds. This will guarantee that (6.3a) and (6.3b) together imply (6.2).

In what follows, we refer to $D_{\mathbf{q}}$ (the constant term in the inequality only containing $\mathbf{q}$-variables) as the $\mathbf{q}$-*term*, and similarly to $D_{\mathbf{r}}$ as the $\mathbf{r}$-*term*.

6.1 Extracting a Cutting Planes Refutation of One of the Subformulas

We now describe how to build the two derivations, one from only restricted $\mathbf{q}$-axioms and the other from only restricted $\mathbf{r}$-axioms, such that each of the derivations has length at most $L(\pi)$, and at least one of the derivations will be deriving contradiction.

The proof is by a forward induction over the cutting planes refutation π. We go through π line by line, maintaining the invariant that for all lines seen so far written on the form (6.1) we have two split inequalities (6.3a) and (6.3b) that satisfy (6.4) and that can be obtained by two independent and valid derivations, one from restricted $\mathbf{q}$-axioms and the other from restricted $\mathbf{r}$-axioms. Moreover every step in π will correspond to at most one step in each of the new derivations, so the length of these derivations will be bounded by the length of π.

Since the final inequality in π is $0 \geq 1$, the split inequalities will be $0 \geq D_{\mathbf{q}}$ and $0 \geq D_{\mathbf{r}}$ with $D_{\mathbf{q}}$ and $D_{\mathbf{r}}$ satisfying $D_{\mathbf{q}} + D_{\mathbf{r}} \geq 1$. Because $D_{\mathbf{q}}$ and $D_{\mathbf{r}}$ are integers, at least one of $D_{\mathbf{q}}$ and $D_{\mathbf{r}}$ must be greater than or equal to 1. This implies that a contradiction has been derived from either restricted $\mathbf{q}$-axioms or restricted $\mathbf{r}$-axioms.

Suppose that all lines so far satisfy the invariant. All we need to show is how to obtain the two split inequalities for the next line while maintaining the invariant. We do so by a case analysis over the cutting planes derivation rules:

Axioms The current line $\sum_{p\in\mathbf{p}} c_p p + \sum_{q\in\mathbf{q}} c_q q + \sum_{r\in\mathbf{r}} c_r r \geq D$ is a $\mathbf{q}$-axiom, an $\mathbf{r}$-axiom or a variable axiom (i.e., $x \geq 0$ or $-x \geq -1$ for some variable x).

Since none of the axioms contains both $\mathbf{q}$- and $\mathbf{r}$-variables we need only consider two cases. If the axiom does not contain $\mathbf{r}$-variables, then it holds that $\sum_{r\in\mathbf{r}} c_r r = 0$ and we can set $D_{\mathbf{q}} = D - \sum_{p\in\mathbf{p}} c_p \rho(p)$ and $D_{\mathbf{r}} = 0$. Note that the inequality with the $\mathbf{r}$-term is $0 \geq 0$. If the axiom contains $\mathbf{r}$-variables, it does not contain $\mathbf{q}$-variables and thus we can set $D_{\mathbf{q}} = 0$ and $D_{\mathbf{r}} = D - \sum_{p\in\mathbf{p}} c_p \rho(p)$. In both cases it is clear that $D_{\mathbf{q}}$ and $D_{\mathbf{r}}$ satisfy (6.4). Moreover, these are valid independent derivation steps, since the split inequalities are either a restricted axiom or the trivial inequality $0 \geq 0$. Therefore the invariant is maintained.

Addition The current line is obtained by adding two inequalities $\sum_{p\in\mathbf{p}} c'_p p + \sum_{q\in\mathbf{q}} c'_q q + \sum_{r\in\mathbf{r}} c'_r r \geq D'$ and $\sum_{p\in\mathbf{p}} c''_p p + \sum_{q\in\mathbf{q}} c''_q q + \sum_{r\in\mathbf{r}} c''_r r \geq D''$.

By the induction hypothesis, we have that the first (respectively, the second) of the above 0–1 linear inequalities is associated to two split inequalities of the form (6.3a) and (6.3b) with $\mathbf{q}$-term $D'_{\mathbf{q}}$ (resp. $D''_{\mathbf{q}}$) and $\mathbf{r}$-term $D'_{\mathbf{r}}$ (resp. $D''_{\mathbf{r}}$), such that

$$D'_{\mathbf{q}} + D'_{\mathbf{r}} \geq D' - \sum_{p\in\mathbf{p}} c'_p \rho(p) \tag{6.5a}$$

$$D''_{\mathbf{q}} + D''_{\mathbf{r}} \geq D'' - \sum_{p\in\mathbf{p}} c''_p \rho(p) \tag{6.5b}$$

hold. By adding the split inequalities separately, we obtain

$$\frac{\sum_{q\in\mathbf{q}} c'_q q \geq D'_{\mathbf{q}} \qquad \sum_{q\in\mathbf{q}} c''_q q \geq D''_{\mathbf{q}}}{\sum_{q\in\mathbf{q}} (c'_q + c''_q) q \geq D'_{\mathbf{q}} + D''_{\mathbf{q}}} \tag{6.6}$$

and

$$\frac{\sum_{r\in\mathbf{r}} c'_r r \geq D'_{\mathbf{r}} \qquad \sum_{r\in\mathbf{r}} c''_r r \geq D''_{\mathbf{r}}}{\sum_{r\in\mathbf{r}} (c'_r + c''_r) r \geq D'_{\mathbf{r}} + D''_{\mathbf{r}}}\,. \tag{6.7}$$

Clearly these are valid derivation steps and, by adding (6.5a) and (6.5b) it is easy to see that the invariant is maintained.

Multiplication The current line is obtained from multiplying $\sum_{p\in\mathbf{p}} c'_p p + \sum_{q\in\mathbf{q}} c'_q q + \sum_{r\in\mathbf{r}} c'_r r \geq D'$ by the positive integer λ.

By the induction hypothesis, we have two split inequalities of the form (6.3a) and (6.3b) with $\mathbf{q}$-term $D'_{\mathbf{q}}$ and $\mathbf{r}$-term $D'_{\mathbf{r}}$, such that such that (6.4) holds. By multiplying each split inequality by λ separately we get

$$\frac{\sum_{q\in\mathbf{q}} c'_q q \geq D'_{\mathbf{q}}}{\sum_{q\in\mathbf{q}} (\lambda c'_q) q \geq \lambda D'_{\mathbf{q}}} \tag{6.8}$$

and

$$\frac{\sum_{r\in\mathbf{r}} c'_r r \geq D'_{\mathbf{r}}}{\sum_{r\in\mathbf{r}} (\lambda c'_r) r \geq \lambda D'_{\mathbf{r}}}, \tag{6.9}$$

and it is easy to see that the invariant is maintained.

Division The current line is obtained from dividing $\sum_{p\in\mathbf{p}} c'_p p + \sum_{q\in\mathbf{q}} c'_q q + \sum_{r\in\mathbf{r}} c'_r r \geq D'$ by λ.

By the induction hypothesis, we have two split inequalities of the form (6.3a) and (6.3b) with $\mathbf{q}$-term $D'_{\mathbf{q}}$ and $\mathbf{r}$-term $D'_{\mathbf{r}}$, such that such that

$$D'_{\mathbf{q}} + D'_{\mathbf{r}} \geq D' - \sum_{p\in\mathbf{p}} c'_p \rho(p) \tag{6.10}$$

holds. By dividing each split inequality by λ separately we get

$$\frac{\sum_{q\in\mathbf{q}} c'_q q \geq D'_{\mathbf{q}}}{\sum_{q\in\mathbf{q}} (c'_q/\lambda) q \geq \lceil D'_{\mathbf{q}}/\lambda \rceil} \tag{6.11}$$

and

$$\frac{\sum_{r\in\mathbf{r}} c'_r r \geq D'_{\mathbf{r}}}{\sum_{r\in\mathbf{r}} (c'_r/\lambda) r \geq \lceil D'_{\mathbf{r}}/\lambda \rceil}. \tag{6.12}$$

Since the coefficients of the $\mathbf{q}$- and the $\mathbf{r}$-variables in the split inequalities are divisible by λ (the original derivation in π was valid), these are valid applications of the division rule. Moreover, by (6.10) we have that

$$\left\lceil \frac{D'_{\mathbf{q}}}{\lambda} \right\rceil + \left\lceil \frac{D'_{\mathbf{r}}}{\lambda} \right\rceil \geq \left\lceil \frac{D'_{\mathbf{q}} + D'_{\mathbf{r}}}{\lambda} \right\rceil \geq \left\lceil \frac{D - \sum_{p\in\mathbf{p}} c'_p \rho(p)}{\lambda} \right\rceil = \left\lceil \frac{D}{\lambda} \right\rceil - \sum_{p\in\mathbf{p}} \frac{c'_p}{\lambda} \rho(p) \tag{6.13}$$

and hence the invariant still holds. (One can note that if we would instead prefer to use general division (2.7) here, this step would still go through).

This completes our goal to produce two independent derivation, one of $0 \geq D_{\mathbf{r}}$ from restricted $\mathbf{r}$-axioms and one of $0 \geq D_{\mathbf{q}}$ from restricted $\mathbf{q}$-axioms, such that $D_{\mathbf{r}} + D_{\mathbf{q}} \geq 1$.

6.2 Writing down the Interpolating Circuit

Our final goal is to build an interpolating circuit for the original formula, and not to produce two independent derivations of the restricted formula. So how can these derivations help us to obtain an interpolant? Observe that since $D_{\mathbf{r}} + D_{\mathbf{q}} \geq 1$ it is enough for the circuit to compute either $D_{\mathbf{r}}$ or $D_{\mathbf{q}}$ in order to output the correct answer, i.e., to output 1 if $D_{\mathbf{r}} \geq 1$ and 0 if $D_{\mathbf{q}} \geq 1$. Indeed, if, for example, the circuit computes $D_{\mathbf{r}}$ and outputs 1 if $D_{\mathbf{r}} \geq 1$ and 0 otherwise, then the fact that $D_{\mathbf{r}} + D_{\mathbf{q}} \geq 1$ guarantees that $D_{\mathbf{q}} \geq 1$ whenever the circuit outputs 0. In the concrete case of the clique-colouring formulas in equations (4.3a)–(4.3e) what this boils down to is that if the circuit outputs 1 the graph defined by ρ has no $(m-1)$-colouring and if it outputs 0 the graph has no m-clique.

Depending on whether $\mathbf{p}$-variables appear only negatively in $\mathbf{r}$-axioms or whether $\mathbf{p}$-variables appear only positively in $\mathbf{q}$-axioms, we build a circuit from the derivation that we obtain from the $\mathbf{r}$-axioms or that from $\mathbf{q}$-axioms. We assume here that $\mathbf{p}$-variables appear only negatively in $\mathbf{r}$-axioms and leave the other case as an exercise to the reader (observing that it requires a slight modification in how we defined the $\mathbf{r}$- and $\mathbf{q}$-terms for lines that are axioms, but this should become clear from the analysis below).

Let us see how to build a circuit based on the derivation we obtain from the $\mathbf{r}$-axioms. The circuit has a gate for every step in the derivation. Each gate receives as input one or two previously computed $\mathbf{r}$-terms and outputs the value of the $\mathbf{r}$-term of the new line derived. For each step in the derivation that is not an axiom, the circuit has a gate that either adds two inputs, or multiplies the input by a positive constant, or divides the input by a positive constant (and rounds up). For axioms, the gate must compute the $\mathbf{r}$-term which is either 0 or $D - \sum c_p \rho(p)$. At the end we will also need a threshold gate that will output 1 if $D_{\mathbf{r}} \geq 1$. Let us see each of these cases in more detail and it will be clear that all gates are monotone.

Axioms with $\mathbf{r}$-variables To compute the value of the $\mathbf{r}$-term of such axioms (either $\mathbf{r}$-axioms or variable axioms for $\mathbf{r}$-variables) we need a circuit that computes $D - \sum c_p \rho(p)$. By assumption, for all axioms that contain $\mathbf{r}$-variables it holds that the coefficients c_p are non-positive. This means that we can compute $D - \sum c_p \cdot \rho(p) = D + \sum(-c_p) \cdot \rho(p)$ with a monotone circuit. Since there are at most w $\mathbf{p}$-variables in any axiom, this means that this subcircuit has at most w gates.

Axioms with no $\mathbf{r}$-variables The value of the $\mathbf{r}$-term in this case is 0, so we use the constant gate 0 (and if we do not want constants in our circuits, it is easy to remove them in a postprocessing step).

Addition The value of the $\mathbf{r}$-term is the sum of two $\mathbf{r}$-terms, so we can use a binary addition gate to compute this sum.

Multiplication The value of the $\mathbf{r}$-term is multiplied by a positive constant, so we can use a unary multiplication gate to compute the new value.

Division The value of the $\mathbf{r}$-term is divided by a positive constant and rounded up, so we can use a unary gate that computes this operation.

Output gate At the end if the $\mathbf{r}$-term is at least 1, then the $\mathbf{r}$-axioms are unsatisfiable and the circuit can output 1. If the $\mathbf{r}$-term is strictly less than 1, then the $\mathbf{q}$-axioms are unsatisfiable and the circuit can output 0. The output can be computed by using a unary threshold gate which outputs 1 if the input is greater or equal to 1.

It is easy to see that all gates described above are monotone, so we can conclude that the circuit is monotone. This completes the proof of Theorem 5.2.

7 Outline of Monotone Real Circuit Lower Bound

Now that we know how to build interpolating circuits from cutting planes refutations of clique-colouring formulas, we want to show that monotone circuits computing interpolants for clique-colouring formulas must have exponential size. Although we will not do present the full proof of this lower bound, let us state the theorem and outline how the argument goes.

Theorem 7.1 ([Pud97]). *Let C be a monotone real circuit that receives as input a Boolean vector of length $\binom{n}{2}$ encoding the edges of a graph on n vertices and let $m = \left\lfloor \frac{1}{8}(n/\log n)^{2/3} \right\rfloor$. If C outputs 1 on all graphs that are not $(m-1)$-colourable and 0 on all graphs that do not contain an m-clique, then C must have size at least* $\exp\left(n^{\Omega(1)}\right)$.

One of the very few known techniques to prove circuit lower bounds is Razborov's approximation method [Raz85]. Razborov's original method, which yields superpolynomial lower bounds for monotone Boolean circuits, was later extended in [AB87] to obtain exponential lower bounds and generalized in [Pud97] to capture monotone real circuits. Let us discuss the basic idea behind this method.

Let $N = \binom{n}{2}$. Given a monotone real circuit C that receives as input a string in $\{0,1\}^N$, we wish to find some class of functions $\mathcal{F} \subseteq \{f : \{0,1\}^N \to \mathbb{R}\}$ such that:

- All dictator functions are in $\mathcal{F}$, that is, for every $i \in [N]$, $\mathcal{F}$ contains the function which on input $(b_1, b_2, \ldots, b_N) \in \{0,1\}^N$ outputs b_i.
- For every gate in C, if it receives as inputs the output of two functions $f_i, f_j \in \mathcal{F}$, then there exists some $\widetilde{f} \in \mathcal{F}$ that closely approximates the output of the gate (i.e., $\widetilde{f}$ errs—disagrees with the output of the gate—only on a small fraction of the inputs).
- The output of C cannot be approximated within the class (i.e., there is no function in the class that only errs—disagrees with the output of the circuit—on a small fraction of the inputs).

Suppose we have such a class of functions $\mathcal{F}$. Then we can argue that $\mathcal{F}$ contains functions that compute the inputs to the bottom gates with no error, for every gate there is a function in $\mathcal{F}$ that introduces very few new errors, but all functions in $\mathcal{F}$ make a huge amount of errors on the circuit output. From this we can conclude that the circuit must have many gates.

What exactly do we mean by errors? To define this concept we introduce the notion of 1-inputs and 0-inputs. We say an input $x \in \{0,1\}^N$ is a 1-input (or 0-input), if the circuit outputs 1 (or 0, respectively) on input x. Given two functions $f, \widetilde{f} : \{0,1\}^N \to \mathbb{R}$, where we think of $\widetilde{f}$ as approximating f, we define to two types of errors:

- If x is a 1-input and $\widetilde{f}(x) < f(x)$, then we say x is a 1-*error* (of $\widetilde{f}$ with respect to f).
- If x is a 0-input and $\widetilde{f}(x) > f(x)$ then we say x is a 0-*error* (of $\widetilde{f}$ with respect to f).

Note that we only consider errors in one direction. The reason for this is that if we want a function f that outputs zero on 0-inputs and one on 1-inputs, it is enough to have a function that is ≤ 0 on 0-inputs and ≥ 1 on 1-inputs, since applying a monotone threshold gate would give us f. So in some sense, if $\widetilde{f}(x) > f(x)$ for x a 1-input or $\widetilde{f}(x) < f(x)$ for x a 0-input, then $\widetilde{f}$ is only "doing better".

To define how much a function $\widetilde{f}$ errors with respect to another function f, we could count errors over all possible inputs, that is, measure errors over the uniform distribution over all inputs. However, the approximation method for proving circuit lower bounds that we are describing allows us to choose any distribution of inputs, whichever suits us best. Once we fix a distribution we must prove that the class of functions $\mathcal{F}$ satisfy the properties we require for this particular distribution.

In our case, we will consider the uniform distribution over *critical* inputs. A 0-input x is critical if flipping the value of any coordinate in x from 0 to 1 yields a 1-input; analogously, a 1-input x is critical if flipping the value of any coordinate in x from 1 to 0 yields a 0-input. For the clique-colouring formula, the critical inputs are graphs of two types: complete $(m-1)$-partite graphs, because these are 0-inputs but adding any edge will ruin the colouring property and change the answer from 0 to 1; and m-cliques (i.e., graphs that contain an m-clique and all vertices not in the clique are isolated vertices), because these are 1-inputs but removing any edge makes it $(m-1)$-colourable.

Let us now sketch in more detail the proof of Theorem 7.1. One first proves that given a monotone real circuit C that receives as input a string in $\{0,1\}^N$, there is a class of functions $\mathcal{F} \subseteq \{f : \{0,1\}^N \to \mathbb{R}\}$ such that:

1. For every $i \in [N]$, the class contains a function which given input $(b_1, b_2, \ldots, b_N) \in \{0,1\}^N$ outputs b_i.
2. If f is the function computed by a gate in C when given as inputs the outputs of two functions $f_i, f_j \in \mathcal{F}$, then there is a function $\widetilde{f} \in \mathcal{F}$ such that there is at most an $\exp(-n^\delta)$ fraction of critical 0-inputs are 0-errors and of critical 1-inputs are 1-errors of $\widetilde{f}$ with respect to f, for some $\delta > 0$.
3. For any function $f \in \mathcal{F}$, either $f \geq 1$ on all critical inputs, or $f < 1$ on $2/9$ fraction of cliques (that is, of critical 1-inputs).

Let us see why the existence of such a class of functions would be enough to prove Theorem 7.1. Item 1 implies that there are functions in $\mathcal{F}$ that approximate the input to the bottom gates of the circuit with no error. From 2 we have that if there are functions in $\mathcal{F}$ that approximate the inputs to a gate, say f_i and f_j, then there is a function in $\mathcal{F}$ that approximates the output of the gate making only a few more errors than f_i and f_j. Finally, by 3 we can conclude that for all $f \in \mathcal{F}$ either $f \geq 1$ for all critical inputs, in which case f is far from approximating the circuit since it errs on all critical 0-inputs, or $f < 1$ for $2/9$ fraction of cliques, in which case f errs on a large fraction of critical 1-inputs and is also far from approximating the circuit. Together this implies that the circuit has many gates in order for the small errors introduced to add up to the large error. To determine how many gates, we must consider two cases.

If $f \geq 1$ on all critical inputs, then the circuit errs on all critical 0-inputs, so there must be at least $\frac{1}{\epsilon} = \exp\left(N^{\delta'}\right)$ gates. If $f < 1$ on $2/9$ fraction of cliques, then there must be at least $\frac{2}{9} \cdot \frac{1}{\epsilon'} = \frac{2}{9} \cdot \exp\left(N^{\delta''}\right)$ gates. In both cases the circuit has exponentially many gates, so this concludes the proof.

From Corollary 5.3 and Theorem 7.1 we immediately get the following corollary.

Corollary 7.2 ([Pud97]). *Cutting planes refutations of the clique-colouring formula* (4.3a)–(4.3e) *for* $m = \left\lfloor \frac{1}{8}(n/\log n)^{2/3} \right\rfloor$ *require length* $\exp\left(n^{\Omega(1)}\right)$.

8 Summing up

In this lecture, we proved that from a short cutting planes refutation of the clique-colouring formula we can construct a small interpolant in the form of a monotone real circuit for the clique-colouring formula. We then sketched the argument (but did not provide a proof) that any such circuit must be of exponential size, and this implies that the cutting planes refutation has exponential length. For the full proof of Theorem 7.1, we will have to refer the reader to Pudlák's paper [Pud97].

References

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

[AR22] Per Austrin and Kilian Risse. Perfect matching in random graphs is as hard as Tseitin. *TheoretiCS*, 1:22:1–22:47, December 2022. Preliminary version in *SODA '22*.

[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. Edmonds 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*.

[Ros97] Arnold Rosenbloom. Monotone real circuits are more powerful than monotone Boolean circuits. *Information Processing Letters*, 61(3):161–164, February 1997.