COMS E6998: Proof Complexity and Applications (Spring'25) February 13, 2025

Lecture 3: Feasible Interpolation

Instructor: *Toniann Pitassi* Scribe: *Steve Marquez*

1 Automatizability

Automatizability is concerned with how efficiently proofs can be found in a given propositional proof system, $\mathcal{P}$. Note that hard-to-refute unsatisfiable CNFs may require superpolynomial-size refutations, and this motivates the following property of a proof system $\mathcal{P}$. Informally $\mathcal{P}$ is *automatizable* if there is an algorithm A such that given as input an unsatisfiable CNF F, A outputs a $\mathcal{P}$-refutation of F in time polynomial in the size of the shortest $\mathcal{P}$ refutation.

Definition 1 (Automatizability)**.** *A proof system $\mathcal{P}$ is (polynomially) automatizable if there exists an algorithm A such that for all unsatisfiable CNF formulas F given as input to A, $A(F)$ outputs a $\mathcal{P}$-refutation of F, and the runtime of A on F is polynomial in the size of the minimum $\mathcal{P}$ proof of F.*

1.1 Remarks

An automatizable proof system is obviously desirable for SAT-solving and automated theorem proving, especially if the proof system in question is fairly strong. It has been shown that tree-like Resolution is automatizable in quasi-polynomial-time, and dag-like Resolution is automatizable in sub-exponential time. We will see later in the course that many algebraic and semialgebraic proof systems (Sherali-Adams, Sum-of-Squares, Nullstellensatz, Polynomial Calculus) are *degree* automatizable, meaning that refutations in these systems can be found in time $n^{O(d)}$ where d is the minimal degree of the refutation. Of these systems, Sum-of-Squares (SOS) is the most powerful, and we will see later that degree-automatizability of SOS has been exploited to give new approximation algorithms for a variety of distributional learning problems.

We note that there is another more relaxed notion of automatizability, called *weak automatizability*. A proof system $\mathcal{P}$ is weakly automatizable if there is a Cook-Reckhow proof system $\mathcal{P}'$ and an algorithm A such that for any unsatisfiable CNF formula F, A returns a $\mathcal{P}'$-refutation of F, and moreover $A(F)$ runs in time polynomial in the size of the shortest $\mathcal{P}$-refutation of F. It is an open problem whether or not Resolution is weakly automatizable. We note that the difference between automatizable and weakly automatizable is analogous to the difference between proper learnability and learnability.

2 Feasible Interpolation

In this section, we define the notion of *feasible interpolation*, which is connected to automatizablity.

Consider an unsatisfiable CNF formula of the form $A(\vec{x}, \vec{z}) \wedge B(\vec{y}, \vec{z})$, where A is a CNF formula over the variables $\vec{x}, \vec{z}$, and similarly B is a CNF formula over the variables $\vec{y}, \vec{z}$. We think of the variables $\vec{z}$ as shared, and the variables $\vec{x}$ (respectively $\vec{y}$) as private to A (B respectively). A CNF formula of this type is called a "split" formula.

Observation 2. *If f is unsatisfiable, then it must be true that for any assignment $\vec{\alpha}$ for $\vec{z}$, either $A(\vec{x},\vec{\alpha})$ or $B(\vec{y},\vec{\alpha})$ must be unsatisfiable.*

Definition 3 (Interpolant Function)**.** *An interpolant function is a function that on input an assignment $\vec{\alpha}$ to the shared variables, says which of the two CNFs, $A(\vec{x},\vec{\alpha})$, or $B(\vec{y},\vec{\alpha})$ is unsatisfiable. Note that both can be unsatisfiable, in which case an interpolant function can give either as a legal answer. This leads to the following definition of an interpolant function associated with a CNF of this form.*

Let $A(\vec{x},\vec{z}) \wedge B(\vec{y},\vec{z})$ be unsatisfiable. An interpolant function, $f_{A\wedge B}$ for this CNF formula takes as input an assignment $\vec{\alpha}$ for $\vec{z}$, and satisfies:

$$f_{A\wedge B}(\vec{\alpha}) = \begin{cases} 1 & \text{if } A(\vec{x},\vec{\alpha}) \text{ is SAT} \\ 0 & \text{if } B(\vec{y},\vec{\alpha}) \text{ is SAT} \\ * & \text{otherwise} \end{cases}$$

Note that in this definition, $f_{A\wedge B}(\vec{\alpha}) = 0$ means that $A(\vec{x},\vec{\alpha})$ must be unsatisfiable. Similarly, when $f_{A\wedge B}(\vec{\alpha}) = 1$, $B(\vec{y},\vec{\alpha})$ has to be unsatisfiable.

Definition 4 (Feasible Interpolation)**.** *$\mathcal{P}$ has feasible interpolation if for every unsatisfiable split formula $A(\vec{x},\vec{z}) \wedge B(\vec{y},\vec{z})$, there is a circuit $C(\vec{\alpha})$ that computes $f_{A\wedge B}$. Furthermore, the size of C is polynomial in the size of the shortest $\mathcal{P}$-refutation of $A \wedge B$.*

Definition 5 (Monotone Feasible Interpolation)**.** *$\mathcal{P}$ has monotone feasible interpolation if for every unsatisfiable split formula $A(\vec{x},\vec{z}) \wedge B(\vec{y},\vec{z})$ where $\vec{z}$ occur only negatively in B and positively in A, then there exists a monotone circuit C that computes $f_{A\wedge B}$ and the size of C is polynomial in the size of the shortest $\mathcal{P}$-refutation of $A \wedge B$.*

3 Automatizability and Feasible Interpolation

Next we relate automatizability and feasible interpolation.

Theorem 6 ([BPR00])**.** *For any propositional proof system $\mathcal{P}$, if $\mathcal{P}$ is automatizable, $\mathcal{P}$ has the feasible interpolation property.*

Proof. Suppose that $A(\vec{x},\vec{z}) \wedge B(\vec{y},\vec{z})$ has a $\mathcal{P}$-refutation π. The circuit C computing the interpolant function $f_{A\wedge B}$ runs as follows:

> On input $\vec{\alpha}$, run the automatization algorithm for $|\pi|$ steps on $A(\vec{x},\vec{\alpha})$. If it outputs a refutation, output 0 (indicating that $A(\vec{x},\vec{\alpha})$ is unsatisfiable); otherwise, output 1 (indicating that $B(\vec{y},\vec{z})$ is unsatisfiable).

The correctness of the algorithm is based on the following observation: if $B(\vec{y},\vec{\alpha})$ is satisfiable, suppose that $\vec{\rho}$ is a satisfying assignment to $\vec{y}$, our split formula under the restriction $\vec{y} \leftarrow \vec{\rho}$ becomes $A(\vec{x},\vec{\alpha}) \wedge B(\vec{\rho},\vec{\alpha}) = A(\vec{x},\vec{\alpha}) \wedge 1$. Therefore, the original proof π restricted to this assignment $\pi|_{\vec{z}\leftarrow\vec{\alpha},\vec{y}\leftarrow\vec{\rho}}$ gives a refutation of $A(\vec{x},\vec{\alpha})$. □

4 Proof System: Cutting Planes

In this section, we study the Cutting Planes proof system, and prove that it has feasible interpolation. Then we will use this property to prove exponential lower bounds for Cutting Planes proofs.

Cutting Planes is a refutation system used for proving unsolvability of systems of linear inequalities where the variables are $x_1, \ldots, x_n \in \mathbb{Z}$.

Definition 7 (Cutting Planes Refutations)**.** *Let $A \in \mathbb{Z}^{m \times n}$ and $b \in \mathbb{Z}^m$. Let $Ax \geq b$ be the system of integer linear inequalities $\{a_1 \cdot x \geq b_1, a_2 \cdot x \geq b_2, \ldots, a_m \cdot x \geq b_m\}$ where $a_1, \ldots, a_m$ are the row vectors of A.*

A Cutting Planes (CP) refutation of $Ax \geq b$ is a finite sequence of inequalities such that each line (inequality) either is one of the original inequality $a_i \cdot x \geq b_i$ or follows from previously derived inequalities by an application of one of the CP inference rules. The goal of CP is to derive the inequality $0 \geq 1$ from the initial set of inequalities, showing that the initial set of inequalities is unsatisfiable over the reals.

The rules are as follows:

***(Division Rule)** From a line $a \cdot x \geq c$ and an interger d dividing every integer in a, we have:*

$$\frac{a \cdot x \geq c}{\frac{a}{d} \cdot x \geq \left\lceil \frac{c}{d} \right\rceil} \quad \text{(Division)}$$

***(Non-Negative Linear Combination Rule)** From two lines $a \cdot x \geq c$ and $b \cdot x \geq d$, we have for any non-negative integers $\alpha, \beta \geq 0$:*

$$\frac{a \cdot x \geq c \qquad b \cdot x \geq d}{(\alpha a + \beta b) \cdot x \geq \alpha c + \beta d} \quad \text{(Linear combination)}$$

Remark 1. *To refute a CNF $F = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, we must first convert each clause to a linear inequality. For example, consider the clause $C = x_1 \vee \neg x_2 \vee x_3$; this converts to the inequality: $x_1 + (1 - x_2) + x_3 \geq 1$. In addition to the inequalities associated with each clause C_i, we additional add $2n$ additional inequalities that force the values of all variables to be in $[0, 1]$: $\{x_i \geq 0, -x_i \geq -1 | i \in [n]\}$.*

Remark 2. *The linear combination rule is sound even over $\mathbb{R}$. However the division rule will preserve only integer solutions.*

Remark 3. *For refuting CNFs, we can assume without loss of generality that all coefficients have magnitude at most 2^{n^2}. Therefore, each line can be encoded using $\mathsf{poly}(n)$ bits, so the size of a refutation can be measured by the number of lines (inequalities) in the CP refutation.*

Remark 4. *One can easily check that CP is sound. We will show in Theorem 8 that it p-simulates Resolution, so is also complete.*

4.1 Cutting Planes vs. Resolution

Cutting Planes is a natural proof system that generalizes Resolution. This is formalized by the next theorem.

Theorem 8. *CP p-simulates Resolution.*

[**Tianqi:** I rewrote this proof. Pls take a look.]

Proof. The idea is to replace every application of the Resolution rule with a CP proof. Recall that a *literal* is either a variable or its negation. For convenience, for any literal p, let $f(p)$ be the function

$$f(p) \triangleq \begin{cases} x_j, & p = x_j \\ 1 - x_j, & p = \neg x_j \end{cases}.$$

For each clause $C = p_1 \vee \cdots \vee p_k$, where $p_1, \ldots, p_k$ are literals, let $f(C) = f(p_1) + \ldots f(p_k)$. Then each such clause is converted to the linear inequality $f(p_1) + \ldots f(p_k) \geq 1$.

Let $C \triangleq p_1 \vee \cdots \vee p_k$, $D \triangleq q_1 \vee \cdots \vee q_l$ where $p_1, \ldots, p_k, q_1, \ldots, q_l$ are all literals. Consider the following application of the Resolution rule:

$$\frac{C \vee x_i \qquad D \vee \neg x_i}{C \vee D} \text{ (Resolution)}$$

We now show that $f(C) + x_i \geq 1, f(D) + 1 - x_i \geq 1 \vdash_{\text{CP}} f(C \vee D) \geq 1$.

The key idea is shown in the example in Figure 1. We assume that there is not variable x_j having positive occurrence in one of C and D and negative occurrence in the other, or the disjunction $C \vee D$ will be trivially valid. Let $C \vee D \triangleq r_1 \vee \ldots r_m$ be the disjunction of C and D with repeated occurrences contracted.

$$\dfrac{\dfrac{\dfrac{x_1 + x_2 + x_3 \geq 1 \qquad x_4 \geq 0}{x_1 + x_2 + x_3 + x_4 \geq 1} \text{ (Addition)} \qquad \dfrac{1 - x_1 + x_2 + x_4 \geq 1 \qquad x_3 \geq 0}{1 - x_1 + x_2 + x_3 + x_4 \geq 1} \text{ (Addition)}}{2x_2 + 2x_3 + 2x_4 \geq 1} \text{ (Addition)}}{x_2 + x_3 + x_4 \geq 1} \text{ (Division)}$$

Figure 1: Example of simulating $\dfrac{x_1 \vee x_2 \vee x_3 \qquad \neg x_1 \vee x_2 \vee x_4}{x_2 \vee x_3 \vee x_4}$ using CP rules.

The first step is to weaken $f(C) + x_i \geq 1$ to $f(C \vee D) + x_i \geq 1$. Note that for each literal p, we have $f(p) \geq 0$ being one of the initial inequalities. So for any literal q_j in D but not in C, we apply the addition rule once with $f(q_j) \geq 0$ to get $f(C) + f(q_j) + x_i \geq 1$. $f(C \vee D) + x_i \geq 1$ can be obtained by repeating this rule.

Similarly, we can weaken $f(D) + 1 - x_i \geq 1$ to $f(C \vee D) + 1 - x_i \geq 1$. Then,

$$\dfrac{\dfrac{f(C \vee D) + x_i \geq 1 \qquad f(C \vee D) + 1 - x_i \geq 1}{2f(C \vee D) \geq 1} \text{ (Addition)}}{f(C \vee D) \geq 1} \text{ (Division)}$$

In order to simulate Resolution with CP, we simply apply the above conversion to each application of the Resolution rule. Since each step can be simulated in CP's by a constant number of steps, it follows that any Resolution refutation can be simulated by a CP refutation of size polynomial in the size of the

Resolution refutation. □

Recall that the pigeon-hole principle requires exponential-size Resolution proofs. The following theorem shows that on the other hand, Resolution cannot p-simulate Cutting Planes.

Theorem 9. *The Pigeonhole Principle (PHP^n_{n-1}) has polynomial-size CP refutations.*

Corollary 10. *Resolution does not p-simulate CP.*

[**Tianqi:** I rewrote this proof. Pls take a look.]

Let us recall the initial clauses for PHP^n_{n-1}:

(Pigeon clauses) $\forall i \in [n] : (P_{i,1} \vee P_{i,2} \vee \ldots \vee P_{i,n-1}$

(Hole clauses) $\forall j \in [n-1] : (P_{1,j} \vee P_{2,j} \vee \ldots \vee P_{n,j})$;
$\forall i \neq i' \in [n], j \in [n-1] : (\neg P_{i,j} \vee \neg P_{i',j})$

The key lemma for proving Theorem 9 is to derive the *hole inequalities* $P_{1,j} + P_{2,j} + \ldots + P_{n,j} \leq 1$ for all $j \in [n-1]$, which means that each hole can only hold at most one pigeon. Similarly we need to derive the *pigeon inequalities* $P_{i,1} \vee P_{i,2} \vee \ldots \vee P_{i,n-1} \geq 1$ for all $i \in [n]$.

Lemma 11. *For each $j \in [n-1]$, the following inequality can be derived from PHP^n_{n-1}:*

$$P_{1,j} + P_{2,j} + \ldots + P_{n,j} \leq 1.$$

Lemma 12. *For each $i \in [n]$, the following inequality can be derived from PHP^n_{n-1}:*

$$P_{i,1} + P_{i,2} + \ldots + P_{i,n-1} \geq 1.$$

We first prove Theorem 9 from Lemmas 11 and 12.

Proof of Theorem 9 from Lemma 11. The proof has 5 steps.

1. Derive the hole inequalities $P_{1,j} + P_{2,j} + \ldots + P_{n,j} \leq 1$ using Lemma 11.
2. Derive the pigeon inequalites using Lemma 12.
3. Sum up all equations from step (1). This will give us
$$\sum_{i \in [n], j \in [n-1]} P_{i,j} \leq n - 1.$$
4. Sum up all equations from step (2). This will give us
$$\sum_{i \in [n], j \in [n-1]} P_{i,j} \geq n.$$
5. Summing up the inequalities we get from steps (3) and (4) gives us $0 \geq 1$.

□

What remains is to prove Lemmas 11 and 12. We prove Lemma 11 below; the proof of Lemma 12 is similar.

Proof of Lemma 11. We prove $P_{1,j}+P_{2,j}+\cdots+P_{n',j} \leq 1$ by induction over n'. When $n'=2$, the inequality is exactly one of the hole clauses. We now show $P_{1,j}+\cdots+P_{n'+1,j} \leq 1$ from $P_{1,j}+\cdots+P_{n',j} \leq 1$ for any $2 \leq n' < n$.

The first step is to sum up $P_{i,j}+P_{n'+1,j} \leq 1$ for all $1 \leq i \leq n'$ and get

$$\sum_{i=1}^{n'} P_{i,j} + n' P_{n'+1,j} \leq n'. \tag{1}$$

Then,

$$\dfrac{\dfrac{\dfrac{\vdots \text{ Equation (1)} \qquad \dfrac{\vdots \text{ (IH)}}{\sum_{i=1}^{n'} P_{i,j} \leq 1}}{\sum_{i=1}^{n'} P_{i,j} + n' P_{n'+1,j} \leq n' \qquad (n'-1)\sum_{i=1}^{n'} P_{i,j} \leq n'-1}\ \text{(Addition)}}{n' \sum_{i=1}^{n'+1} P_{i,j} \leq 2n'-1}\ \text{(Multiplication)}}{\sum_{i=1}^{n'+1} P_{i,j} \leq 1}\ \text{(Division)}$$

□

4.2 Exponential Lower Bounds for CPs using Feasible Interpolation

In this section we prove an exponential lower bounds for CP using feasible interpolation.

The overview of the proof is as follows. First, we show that CPs has monotone feasible interpolation. In particular, for any unsatisfiable split formula, $F = A(\vec{x},\vec{z}) \wedge B(\vec{y},\vec{z})$ we have:

1. If F has a polynomial-size CP refutation where all coefficients are small (the length of all coefficients in binary is $O(\mathsf{polylog}(n))$, then there is a polynomial-size monotone circuit for computing the interpolant function;
2. And if F has a polynomial-size CP refutation without any bound on the coefficient length (recall that the coefficient can be encoded by at most $\mathsf{poly}(n)$ bits without loss of generality), then there is a polynomial-size monotone *real* circuit for computing the interpolant function.

A monotone real circuit is a generalization of a monotone circuit, given by the following definition. We note that to get the main idea behind the CP lower bound, the reader should focus on the simpler case where the coefficients are bounded in length (case 1 above).

Definition 13 (Monotone Real Circuit)**.** *A monotone real circuit for $f : \{0,1\}^n \to \{0,1\}$ is a sequence of functions $g_1,\ldots,g_s$ where $g_s = f$ and for all $i \leq s$, g_i satisfies at least one of these conditions: $g_i = x_i$ or there is a monotone real function $\Phi : \mathbb{R}\times\mathbb{R} \to \mathbb{R}$, and $j,k < i$ such that $g_i = \Phi(g_j, g_k)$.*

We prove below the special case corresponding to (1) above. Consider an unsatisfiable split CNF formula $F = A(\vec{x}, \vec{z}) \wedge B(\vec{y}, \vec{z})$ that is monotone with respect to $\vec{z}$. Let π be a CP refutation of F, where all coefficients in π have length at most $\mathsf{polylog}(n)$. Given an assignment $\vec{\alpha}$ to the $\vec{z}$ variables, we want to show that there exists a monotone circuit $C(\vec{\alpha})$, of size polynomial in the size of π, that outputs $f_{A \wedge B}(\vec{\alpha})$.

Theorem 14. *There is a polynomial-sized circuit $C(\vec{\alpha})$ that outputs $f_{A \wedge B}(\vec{\alpha})$.*

Proof. Let $\vec{\alpha}$ be an assignment to the variables $\vec{z}$. We want to show that for each line $f(\vec{x})+g(\vec{y})+h(\vec{\alpha}) \geq D$ in π, there exists D_0 and D_1, where $D_0 + D_1 \geq D - h(\vec{\alpha})$ such that, in Cutting Planes, we can derive $f(\vec{x}) \geq D_0$ from $A(\vec{x}, \vec{\alpha})$ and $g(\vec{y}) \geq D_1$ from $B(\vec{y}, \vec{\alpha})$. We prove this by induction on the size of the proof.

(Base case) This is trivially true for the initial clauses (when the proof has size 1).

(Linear combination) Suppose that the j^{th} line of the proof is derived from the following two previously derived inequalities by the linear combination rule

$$\frac{f_1(\vec{x}) + g_1(\vec{y}) + h_1(\vec{\alpha}) \geq C \qquad f_2(\vec{x}) + g_2(\vec{y}) + h_2(\vec{\alpha}) \geq D}{\beta\left(f_1(\vec{x}) + g_1(\vec{y}) + h_1(\vec{\alpha})\right) + \gamma\left(f_2(\vec{x}) + g_2(\vec{y}) + h_2(\vec{\alpha})\right) \geq \beta C + \gamma D} \quad \text{(Linear combination)}$$

By the induction hypothesis, we have the following linear inequalities

$$\begin{aligned} f_1(\vec{x}) \geq C_0, \quad g_1(\vec{y}) \geq C_1 \quad \text{where} \quad C_0 + C_1 \geq C - h_1(\vec{\alpha}), \\ f_2(\vec{x}) \geq D_0, \quad g_2(\vec{y}) \geq D_1 \quad \text{where} \quad D_0 + D_1 \geq D - h_2(\vec{\alpha}), \end{aligned}$$

where $f_1(\vec{x}) \geq C_0$ and $f_2(\vec{x}) \geq D_0$ are derivable from $A(\vec{x}, \vec{\alpha})$, and $g_1(\vec{y}) \geq C_1$ and $g_2(\vec{y}) \geq D_1$ are derivable from $B(\vec{y}, \vec{\alpha})$.

From these inequalities, we can simply apply an addition rule and derive $\beta f_1(\vec{x}) + \gamma f_2(\vec{x}) \geq \beta C_0 + \gamma D_0$ and $\beta g_1(\vec{y}) + \gamma g_2(\vec{y}) \geq \beta C_1 + \gamma D_1$.

(Division) Suppose that the j^{th} line of the proof was derived via the division rule

$$\frac{f(\vec{x}) + g(\vec{y}) + h(\vec{\alpha}) \geq D}{\frac{1}{d}\left(f(\vec{x}) + g(\vec{y}) + h(\vec{\alpha})\right) \geq \left\lceil \frac{D}{d} \right\rceil} \quad \text{(Division)}$$

By the induction hypothesis, there exists D_0 and D_1 such that $D_0 + D_1 \geq D - h(\vec{\alpha})$ and $f(\vec{x}) \geq D_0$ and $g(\vec{y}) \geq D_1$ are derivable from $A(\vec{x}, \vec{\alpha})$ and $B(\vec{y}, \vec{\alpha})$, respectively.

Then we can apply the division rule to each of the inequalities and get

$$\frac{1}{d} f(\vec{x}) \geq \left\lceil \frac{D_0}{d} \right\rceil \qquad \frac{1}{d} g(\vec{y}) \geq \left\lceil \frac{D_1}{d} \right\rceil$$

where

$$\left\lceil \frac{D_0}{d} \right\rceil + \left\lceil \frac{D_1}{d} \right\rceil \geq \left\lceil \frac{D_0 + D_1}{d} \right\rceil \geq \left\lceil \frac{D - h(\vec{\alpha})}{d} \right\rceil = \left\lceil \frac{D}{d} \right\rceil - \frac{h(\vec{\alpha})}{d}.$$

The last equality follows from the fact that d divides all coefficients in $h(\vec{\alpha})$.

Thus we can obtain CP derivations of $0 \geq D_0$ from $A(\vec{x}, \vec{\alpha})$ and $0 \geq D_1$ from $B(\vec{y}, \vec{\alpha})$. Since $D_0 \geq 1$ or $D_1 \geq 1$, we have a refutation $0 \geq 1$ from either $A(\vec{x}, \vec{\alpha})$ or $B(\vec{y}, \vec{\alpha})$. So our algorithm A computes $D_0 + D_1$ and outputs 0 if $D_0 \geq 0$ and 1 if $D_1 \geq 1$. □

4.2.1 Lower Bounds for the Clique-Coclique Formula

Let us use everything we have learned so far and apply it to the Clique-Coclique formula.

Definition 15 (**Clique**). *In graph theory, a clique is a subset of vertices within a graph where every single vertex is directly connected to every other vertex in that subset.*

Definition 16 (**Coclique**). *A coclique is a set of vertices in a graph where no two vertices are connected by an edge.*

The Clique-Coclique formula is a split CNF formula defined as: $F = Clique_k(\vec{x}, \vec{z}) \wedge Color_{k-1}(\vec{y}, \vec{z})$. Here, $\vec{z}$ encodes an undirected graph $G = (V, E)$ where $|V| = n$. In particular, for any $1 \leq i < j \leq n$, $z_{i,j} = 1$ if and only if $(i, j) \in E$. For simplicity, we let $z_{i,j} = z_{j,i}$ be the same variable. $\vec{x}$ is a $k \times n$ matrix encoding a clique of size k. Semantically, each row has exactly one 1, and the set of columns having 1's form the clique. $\vec{y}$ is a $(k-1) \times n)$ matrix, where $yi, u = 1$ if and only if the vertex u is assigned with the i-th color. The two formulas are formally defined as the following.

Definition 17 ($Clique_k(\vec{x}, \vec{z})$). *$Clique_k(\vec{x}, \vec{z})$ is the conjunction of the following constraints:*

(1) $\forall i \in [k] : \bigvee_{v=1}^{n} x_{i,v}$

(2) $\forall i \neq j \in [k], \forall v \in [n] : \neg x_{i,v} \vee \neg x_{j,v}$

(3) $\forall u \neq v \in [n], \forall i \neq j \in [k] : x_{i,u} \vee x_{j,v} \rightarrow z_{u,v}$

Constraint (1) *and* (2) *says that $\vec{x}$ defines a subset $S \subseteq [n]$ of size k. Constraint* (3) *says that all edges between clique vertices are in G.*

Definition 18 ($Color_{k-1}(\vec{y}, \vec{z})$). *$Color_{k-1}(\vec{y}, \vec{z})$ is defined by the following constraints:*

(1) $\forall u \in [n] : \vee_{v=1}^{n} y_{i,v}$

(2) $\forall i \neq j \in [k-1], \forall u \in [n] : \neg y_{i,u} \vee \neg y_{j,u}$

(3) $\forall u \neq v \in [n], \forall i \in [k-1] : \neg y_{i,u} \vee \neg y_{i,v} \vee \neg z_{u,v}$

Constraints (1) *and* (2) *say that $\vec{y}$ represents a legal coloring of n. Constraint* (3) *says that there is no edge in G between vertices of the same color.*

By construction we can define the following interpolant function:

$$f(\vec{z}) = \begin{cases} 1 & \text{if graph encoded by } \vec{z} \text{ contains a } k\text{-clique} \\ 0 & \text{if graph encoded by } \vec{z} \text{ has a } k-1 \text{ coloring} \\ * & \text{otherwise} \end{cases}$$

The Clique-Coclique formula has two properties: minterm and maxterm, which will be used to calculate the lower bounds of the formula.

Definition 19. *For a fix k, a minterm is a graph containing a k-clique and no other edges. A maxterm is a maximal graph that is $(k-1)$ colorable such that its vertices are partitioned into $k-1$ groups and $E_{i,j} = 1$ if and only if vertices i and j belong to different groups.*

Theorem 20 ([RAZ85])**.** *Let $k = \sqrt{n}$. Based on the construction of the formula and its min/max term properties, any monotone circuit C that accepts all minterms and rejects all maxterms has* $\text{size}(2^{\Omega(n^\epsilon)})$ *for some $\epsilon > 0$.*

We now combine the lower bound in Theorem 20 and the feasible interpolation of CP in Section 4.2 to get an exponential lower bound for CP.

Theorem 21. *Any CP refutation of the Clique-Coclique formula requires exponential size.*

Proof. By monotone feasible interpolation for CP, a size(s) CP refutation implies a poly(s) size monotone real circuit for separating k-cliques from $(k-1)$ colorable graphs for all k-values. However, by Razborov [RAZ85] (for the low coefficient cases) and [HC99] (for the more general case of coefficients of length $\mathsf{poly}(s)$), monotone circuits for the clique-coclique formula requires size at least $\exp(n^\epsilon)$. For $k = \sqrt{n}$, this implies a $2^{\Omega(n^\epsilon)}$ lower bound on the size of CP refutations. □

5 Final Remarks and Further Research

For a long time the only lower bound for Cutting Planes was via monotone feasible interpolation, and therefore only for split-form formulas. However, recent work by [FPPR22] and [HP17] proved exponential lower bounds for random k-CNFs for $k = O(\log n)$, by generalizing the feasible interpolation method for arbitrary formulas. This leads to the open question of whether we can improve the lower bounds for random kCNFs for $k = O(1)$.

We note that for Resolution, exponential lower bounds were proven for random k-CNFs for $k = O(1)$ by Chvátal and Szemerédi [CS88].

References

[BPR00] Maria Luisa Bonet, Toniann Pitassi, and Ran Raz. "On Interpolation and Automatization for Frege Systems". In: *SIAM Journal on Computing* 29.6 (2000), pp. 1939–1967. DOI: 10.1137/S0097539798353230. URL: https://doi.org/10.1137/S0097539798353230 (cit. on p. 2).

[CS88] Vasek Chvátal and Endre Szemerédi. "Many Hard Examples for Resolution". In: *J. ACM* 35.4 (1988), pp. 759–768. DOI: 10.1145/48014.48016. URL: https://doi.org/10.1145/48014.48016 (cit. on p. 9).

[FPPR22] Noah Fleming, Denis Pankratov, Toniann Pitassi, and Robert Robere. "Random Θ(log n)-CNFs are Hard for Cutting Planes". In: *J. ACM* 69.3 (2022), 19:1–19:32. DOI: 10.1145/3486680. URL: https://doi.org/10.1145/3486680 (cit. on p. 9).

[HC99] Armin Haken and Stephen A. Cook. "An Exponential Lower Bound for the Size of Monotone Real Circuits". In: *J. Comput. Syst. Sci.* 58.2 (1999), pp. 326–335. DOI: 10.1006/JCSS.1998.1617. URL: https://doi.org/10.1006/jcss.1998.1617 (cit. on p. 9).

[HP17] Pavel Hrubes and Pavel Pudlák. "Random Formulas, Monotone Circuits, and Interpolation". In: *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*. Ed. by Chris Umans. IEEE Computer Society, 2017, pp. 121–131. DOI: 10.1109/FOCS.2017.20. URL: https://doi.org/10.1109/FOCS.2017.20 (cit. on p. 9).

[RAZ85] A. A. Razborov. “Lower bounds on the monotone complexity of some Boolean functions”. In: *Dokl. Akad. Nauk SSSR* 281.4 (1985), pp. 798–801. URL: http://mi.mathnet.ru/dan9192 (cit. on p. 9).