Boolean Satisfiability
Lecture B:
An exponential-size lower bound for Cutting Planes.
Short Frege proofs for PHP

Edward A. Hirsch*

June 9, 2026

Lecture B

In this lecture we prove an exponential-size lower bound on the size of Cutting Planes proofs of clique-coloring formulas and sketch polynomial-size upper bounds on the size of Frege proofs of PHP.

Contents

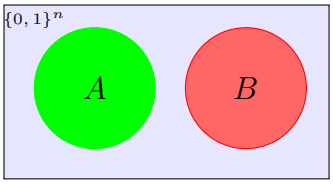
*Ariel University, http://edwardahirsch.github.io/edwardahirsch

1 An exponential lower bound on the length of Cutting Planes proofs

1.1 Interpolation by circuits

Consider two disjoint sets $A, B \in \mathbf{NP}$. As $A \cap B = \emptyset$, we can separate them by a function (called an **interpolant**) that gives the answer 0 on A, gives the answer 1 on B, and can give any answer on the complement of $A \cup B$:

- $x \in A \Rightarrow C(x) = 1$.
- $x \in B \Rightarrow C(x) = 0$.
- $x \notin (A \cup B)$ — doesn't matter.



We will be interested in presenting this function as a Boolean circuit. Is there a polynomial-size circuit C achieving this goal? If we think about cryptographic encoding of single bits, then sets $A = \{\textit{codewords for } 0\}$ and $B = \{\textit{codewords for } 1\}$ should not be distinguishable this way. Yet nobody so far can prove that there are **NP**-sets for which it is impossible.

Example 1. $A =$ {satisfiable formulas in CNF}, $B =$ {unsatisfiable formulas in CNF that have resolution refutations of size at most $9n^9$}.

Let us consider a simpler setting: **monotone** circuits. These are circuits that use only monotonically increasing operations: for example, $\vee$ and $\wedge$ (if an argument increases, then the value does not decrease).

Is it reasonable to separate **NP**-sets with a monotone circuit? Not always: if the sets themselves are not "monotone", it's weird. However, for monotone sets it is reasonable. For example, if you add an edge to a graph (increase the value of the corresponding variable), then you may acquire a new m-clique, but you cannot lose any m-cliques you had in your graph.

Replace A and B by their characteristic functions ($A(x) = 1 \iff x \in A$). Let A be the set of graphs containing an m-clique, so the "clique function" $A(x)$ that tells us whether the graph x contains an m-clique, is monotonically increasing (as as a function of the $\binom{n}{2}$ variables corresponding to the edges of an n-vertex graph). On the other hand, the "coloring function" $B(x)$ that tells us whether x can be colored in $m-1$ colors, is monotonically decreasing.

Fortunately, in this model exponential-size lower bounds are known:

Theorem 1 (Razborov; Alon–Boppana)**.** *Let* $m = \lfloor \frac{1}{8}(n/\log n)^{2/3} \rfloor$,
$A = \{n\textit{-vertex graphs containing } m\textit{-cliques}\}$, $B = \{n\textit{-vertex } (m-1)\textit{-colorable graphs}\}$.
For every monotone Boolean circuit distinguishing A *and* B,

$$|C| = 2^{\Omega(\sqrt{m})}.$$

Pavel Pudlák generalized this theorem to circuits that operate with integer numbers (and not just bits) and use monotone integer operations (addition, multiplication by a nonnegative integer constant, division by a nonnegative integer constant with rounding up).

1.2 Interpolation in logic

A similar notion is known in logic:

Theorem 2 (Craig)**.** *If* $A(\vec{p},\vec{q}) \Rightarrow B(\vec{p},\vec{r})$*, then one can construct a formula* $C(\vec{p})$ *such that* $A(\vec{p},\vec{q}) \Rightarrow C(\vec{p})$ *and* $C(\vec{p}) \Rightarrow B(\vec{p},\vec{r})$*.*

We are interested in the propositional version of this theorem, so A, B, C are Boolean formulas (and C can be large!). Let us rewrite this theorem:

Theorem 3. *If* $A(\vec{p},\vec{q}) \wedge B(\vec{p},\vec{r})$ *is unsatisfiable, then one can construct a formula* $C(\vec{p})$ *such that* $\neg C(\vec{p}) \Rightarrow \neg A(\vec{p},\vec{q})$ *and* $C(\vec{p}) \Rightarrow \neg B(\vec{p},\vec{r})$*.*

Let A, B be in the de Morgan basis. If p_i's occurs only positively in A and only negatively in B, then one construct a monotone $C(\vec{p})$, which we will do. An example of appropriate formulas A and B is given by clique-coloring formulas:

$\bigvee_{i=1}^{n} q_{ki}$	for $1 \leqslant k \leqslant m$	the k-th node of the clique
$\overline{q_{ki}} \vee \overline{q_{k'i}}$	for $1 \leqslant i \leqslant n$ and $1 \leqslant k < k' \leqslant m$	all nodes are different
$\overline{q_{ki}} \vee \overline{q_{k',j}} \vee \boldsymbol{p_{ij}}$	for $1 \leqslant k < k' \leqslant m$ and $1 \leq i < j \leq n$	edge between the nodes k and k'
$\bigvee_{\ell=1}^{m-1} r_{i\ell}$	for $1 \leqslant i \leqslant n$	vertex i is colored
$\overline{\boldsymbol{p_{ij}}} \vee \overline{r_{i\ell}} \vee \overline{r_{j\ell}}$	for $1 \leq i < j \leq n$, $1 \leqslant \ell \leqslant m-1$	correctness of the coloring

1.3 An exponential lower bound for Clique-Coloring formulas

Our proof strategy will be

- Construct a monotone circuit C based on a Cutting Planes refutation so that the circuit size is at most polynomial w.r.t. the refutation size.
- Conclude that if there is a short CP proof, then there is a small C.
- Get a contradiction with Razborov's theorem (Pudlák's version).

The structure of the circuit will mimic the structure of the proof dag.

General construction. Our circuit is given a specific graph (that is, p_{ij}'s values) on its input, and will answer 0, if the graph contains no m-clique, and 1, if the graph is not $(m-1)$-colorable. (If the graph contains no clique and is not colorable, then the circuit can answer anything.) Namely, the circuit will tell us which of the two parts of clique-coloring formula is unsatisfiable.

Consider the Cutting Planes refutation. Substitute these values for p_{ij}'s into it.

The input inequalities contain now only q_{ki}'s or only $r_{j\ell}$'s. There are no mixed inequalities. Each inequality appearing in the refutation will be split into a q-part and an r-part:

$$\sum_{k,i} \alpha_{ki} q_{ki} + \sum_{i,j} \beta_{ij} p_{ij} + \sum_{j,\ell} \gamma_{j\ell} r_{j\ell} \geq c \longrightarrow \begin{array}{rcl} \sum_{k,i} \alpha_{ki} q_{ki} & \geq & c' \\ \sum_{j,\ell} \gamma_{j\ell} r_{j\ell} & \geq & c'' \end{array}$$

How to split $c = c' + c''$? The two inequalities should be at least as strong. It suffices to have $c' + c'' \geq c - \sum_{ij} \beta_{ij} p_{ij}$. Indeed, then they are at least as strong: otherwise there are values such that

$$\begin{array}{rcl} \sum_{k,i} \alpha_{ki} q_{ki} + \sum_{i,j} \beta_{ij} p_{ij} + \sum_{j,\ell} \gamma_{j\ell} r_{j\ell} & < & c \\ \sum_{k,i} \alpha_{ki} q_{ki} & \geq & c' \\ \sum_{j,\ell} \gamma_{j\ell} r_{j\ell} & \geq & c'' \end{array}$$

But then

$$c' + c'' < c - \sum_{ij} \beta_{ij} p_{ij}.$$

Splitting the constants. We will split the constants in every derived inequality. We will do it step by step starting from the input inequalities.

Input inequalities are already split (there is only one part). How could the next inequality be obtained? There are three cases.

Addition. Let P_a, Q_a, R_a denote the corresponding sums $\sum \beta_{ij}^{(a)} p_{ij}$, etc. We had a derivation step

$$\frac{P_1 + Q_1 + R_1 \geq c_1 \ , \ P_2 + Q_2 + R_2 \geq c_2}{(P_1 + P_2) + (Q_1 + Q_2) + (R_1 + R_2) \geq c_1 + c_2}$$

The two premises already were split as

$$\left[\begin{array}{rcl} Q_1 & \geq & c_1' \\ R_1 & \geq & c_1'' \end{array} \right] \text{ and } \left[\begin{array}{rcl} Q_2 & \geq & c_2' \\ R_2 & \geq & c_2'' \end{array} \right].$$

Let us simply take the sum of the constants, that is, split the conclusion into

$$\left[\begin{array}{rcl} Q_1 + Q_2 & \geq & c_1' + c_2' \\ R_1 + R_2 & \geq & c_1'' + c_2'' \end{array} \right].$$

We need that

$$(c_1' + c_2') + (c_1'' + c_2'') \geq (c_1 + c_2) - (P_1 + P_2).$$

Given

$$\begin{array}{rcl} c_1' + c_1'' & \geq & c_1 - P_1, \\ c_2' + c_2'' & \geq & c_2 - P_2 \end{array}$$

it is obvious.

Multiplication by a nonnegative constant is similar.

The rounding rule. If the next inequality is obtained using the rounding rule, let us divide and round the constants accordingly:

$$\frac{d \cdot P + d \cdot Q + d \cdot R \geq c \qquad \left[\begin{array}{lcl} d \cdot Q & \geq & c' \\ d \cdot R & \geq & c'' \end{array}\right]}{P + Q + R \geq \lceil c/d \rceil \qquad \left[\begin{array}{lcl} Q & \geq & \lceil c'/d \rceil \\ R & \geq & \lceil c''/d \rceil \end{array}\right]}$$

Since

$$c' + c'' \geq c - d \cdot P,$$

we have

$$\lceil c'/d \rceil + \lceil c''/d \rceil \geq \lceil c/d \rceil - P.$$

Putting everything together: the circuit. The end of the CP refutation is split as

$$0 \geq 1 \longrightarrow \left[\begin{array}{lcl} 0 & \geq & c' \\ 0 & \geq & c'' \end{array}\right]$$

We know that

$$c' + c'' \geq 1.$$

Therefore, one of the inequalities in this pair is a contradiction showing us what input part was contradictory ("Q" or "R").

The circuit (it is enough to build it for the "R" part can be built from the end (that is, from its output associated with the last (contradictory) node of the proof). It is based on the proof graph, and the constants c' and c'' are computed from the previous level constants.

The circuit inputs are p_{ij}'s, and the first constants are computed simply by substituting their values in the input inequalities (that is, computing the relevant sums $\sum \beta_{ij} p_{ij}$ and subtracting them from the constants of the input inequalities).

(See the picture in Fig. 1.)

Also there are axioms

$$0 \leq p_{ij} \leq 1$$

As they do not contain q or r-variables, We can treat them as the first and the second inequalities in the pairs, respectively (according to the sign of p_{ij}). (The other inequality in such a pair is $0 \geqslant 0$.)

As we see from our construction, the constants are computed using monotone operations: the addition, the multiplication by a nonnegative constant, the division with rounding up. (Also, to make it Boolean, formally the last gate is the sign gate "if $c'' > 0$, then 1, else 0", which is also monotone. Here c'' is the last computed constant: for the R part of the contradiction $0 \geqslant 1$.) So our circuit is monotone, and the number of its internal gates is the same as the number of steps in the CP refutation.

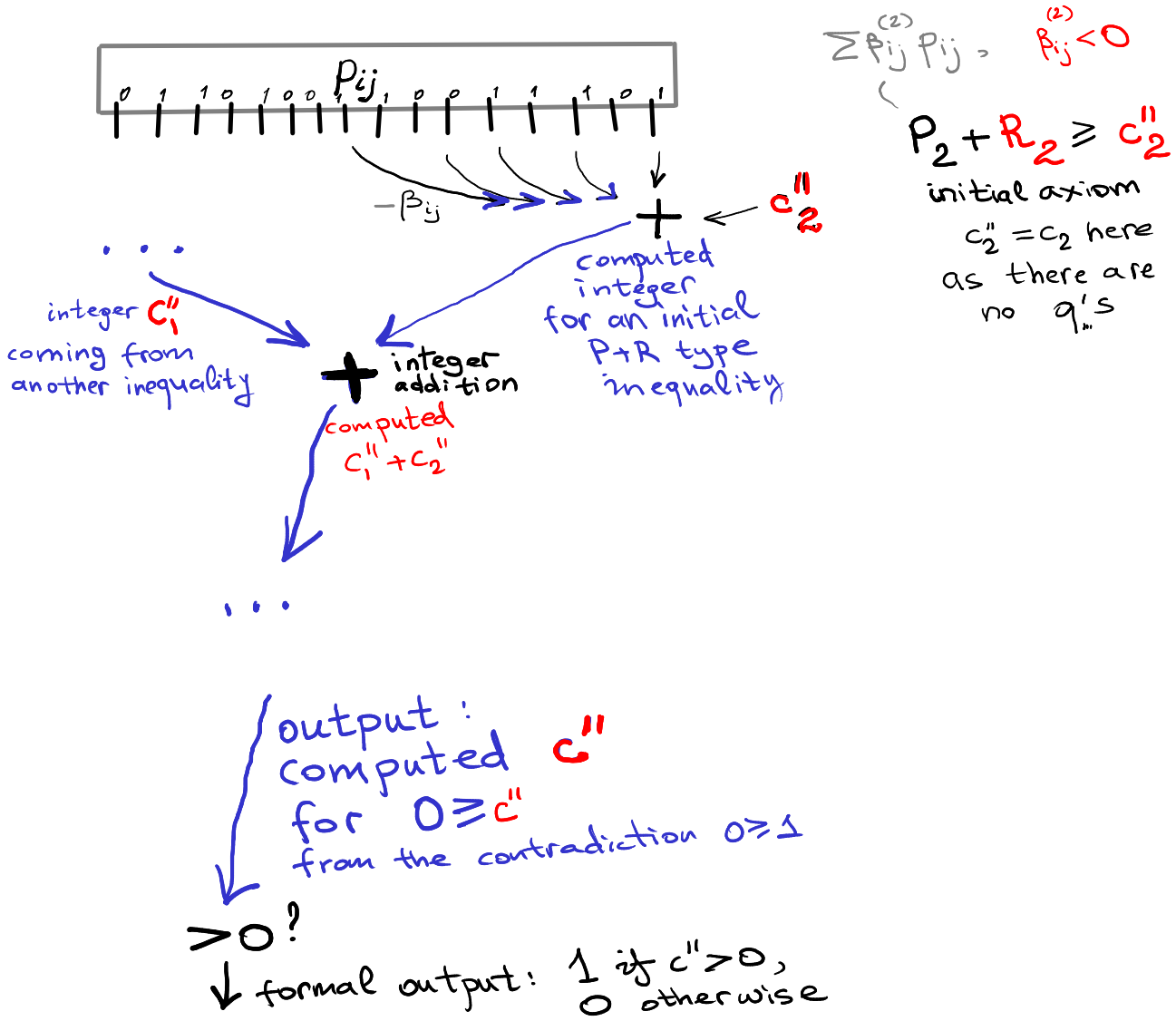


Figure 1: Interpolating circuit saying whether the coloring part is contradictory.

2 PHP in Frege

Theorem 4 (Buss)**.** *PHP has polynomial-size proofs in Frege systems.*

Sketch. Why does PHP have short CP proofs? Recall the CP refutation for it: we compute a big sum there, so the reason is: CP can count.

Consider an alternative refutation of PHP in Extended Frege. Since we can introduce new variables for the gates of any circuits, we can encode any **Boolean** circuits including integer addition, multiplication, rounding (working on binary representations of integer numbers).

In Frege systems, we cannot encode any circuits. However, we can use formulas. A log-depth circuit can be transformed into a formula with a polynomial increase in size.

So it remains to implement integer operations using log-depth circuits to express the formula $\mathrm{Count}_i(x_1,\dots,x_n)$ = the i-th bit of $\sum_{j=1}^{n} x_j$. and to learn how to work with these formulas.

Unsuccessful try. We first try to use plain carry-save addition to construct Count_i:

$$\begin{aligned} Add_i(\vec{y},\vec{z}) &= y_i \oplus z_i \oplus Carry_i \\ Carry_i(\vec{y},\vec{z}) &= \bigvee_{j<i}\left(y_j \wedge z_j \wedge \bigwedge_{j<k<i} (y_k \oplus z_k)\right) \end{aligned}$$

However, if we use these definitions recursively to construct the sum of the "$n\times(n-1)$ matrix", we would get formulas of superlogarithmic depth and thus superpolynomial size (superpolynomial number of occurrences of x_i). However, there are more efficient implementations of integer addition.

The 3-into-2 approach.

Fact (and appetizer for the **Parallel Algo** course):

Addition and even multiplication of n-bit numbers can be made $\log n$ depth

Using these formulas would be too technical here and would be an overkill: our numbers are small enough.

However, we'll use one idea from those parallel algorithms: namely, "adding 3-into-2". That is, to avoid sequential computing (and thus deep formulas for) of the carries, we compute the sum of 3 integers and "lazily" store the result in 2 integers thus avoiding propagating these carries. So we turn three numbers (bit strings) $\vec{a},\vec{b},\vec{c}$ into two $\vec{d},\vec{e}$ so that $a+b+c=d+e$. If we start with n integers and decrease their number by a factor of 1.5 at each 3-into-2 step, in $\log_{3/2} n$ steps we will obtain just two numbers.

In fact, we'll do even "4-into-2" (by combining two 3-into-2 steps). We define CSA, which will be a function of four arguments returning **two** results. We define it inductively up to $i \sim \log n$:

$$\begin{aligned}(\mathbf{S^{0,j}}, \mathbf{C^{0,j}}) &= (x_j, 0)\\ (\mathbf{S^{i+1,j}}, \mathbf{C^{i+1,j}}) &= CSA(CSA(C^{i,2j}, S^{i,2j}, C^{i,2j+1}), S^{i,2j+1})\end{aligned}$$

Of course, we need to write a formula for each bit separately (not for a pair).

Claim: $C^{ij} + S^{ij} = \sum_{j\cdot 2^i \leqslant k < (j+1)\cdot 2^i} x_k$ (it is easy to see by induction on i).

Now starting from $(\overline{p}_{a,1} \vee \overline{p}_{b,1})$ we can inductively prove that

- hole 1 contains at most 1 pigeon (i.e., $\sum_{a=1}^{n} p_{a,1} \leqslant 1$),
- holes 1,2 contain at most 2 pigeons (i.e., $\sum_{a=1}^{n}(p_{a,1} \vee p_{a,2}) \leqslant \sum_a p_{a,1} + \sum_b p_{b,2} \leqslant 2$),

$\ldots$,

- $n-1$ holes contain at most $n-1$ pigeons (i.e., $\sum_{a=1}^{n}(p_{a,1} \vee \ldots \vee p_{a,n-1}) \leqslant n-1$).

and

there are at least n pigeons hosted in some hole (i.e., $\sum_{a=1}^{n}(p_{a,1} \vee \ldots \vee p_{a,n-1}) \geqslant n$).

We will not elaborate on these technical details, see the original paper [Bus87] or its later explanation [Bus97, Theorem 8]. □

3 Takeaway

Efficient monotone interpolation and (generalized) Razborov's theorem allowed us to show exponential-size lower bounds on CP proofs of clique-coloring formulas.

Frege systems can count, and thus can prove PHP within polynomial size.

Historical notes and further reading

Jan Krajíček suggested to use monotone interpolation for proving exponential-size lower bounds on the length of Cutting Planes proofs and proved such bounds for CP proofs with small coefficients. Pavel Pudlák [Pud97] generalized his ideas by generalizing Alexander Razborov's theorem to integer arithmetic and showed the bound that we proved in the lecture (you can check the original paper for more details).

The short proof of PHP in Frege systems is due to Sam Buss [Bus87] (again, you can check the original paper or a survey below for the details).

References

[Bus87] Samuel R. Buss. Polynomial size proofs of the propositional pigeonhole principle. Journal of Symbolic Logic, 52:916–927, 1987.

[Bus97] Samuel R. Buss. Propositional Proof Complexity: An Introduction. https://mathweb.ucsd.edu/~sbuss/ResearchWeb/marktoberdorf97/paper.pdf

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