

PROOF COMPLEXITY AS A COMPUTATIONAL LENS

LECTURE 4

Recap

- Resolution proof system
- lower bounds for PHP and Tseitin formulas (handshaking lemma)
- Analyze resolution proofs using Prosecutor - Defendant game
 - Efficient Prosecutor strategies
 - $\Leftrightarrow$ short resolution proofs
 - Defendant strategies that force Prosecutor to write down lots of different info
 - $\Leftrightarrow$ lower bounds on resolution proof length
- Prosecutor - Defendant game is CHARACTERIZATION of resolution, so good Defendant strategy should always exist for hard formulas
- But there might be easier ways to prove lower bounds:
 - lower bounds on resolution width (size of clauses)
 - reduce to circuit complexity via (feasible) interpolation

Today we will talk about interpolation

BOOLEAN CIRCUIT

Directed acyclic graph (DAG) C_n
 n sources (nodes without incoming edges)
 unique sink (without outgoing edges)
 Sources labelled by variables $x_1, \dots, x_n$
 Non-source vertices have in-degree / fan-in
 1 or 2 and are called gates

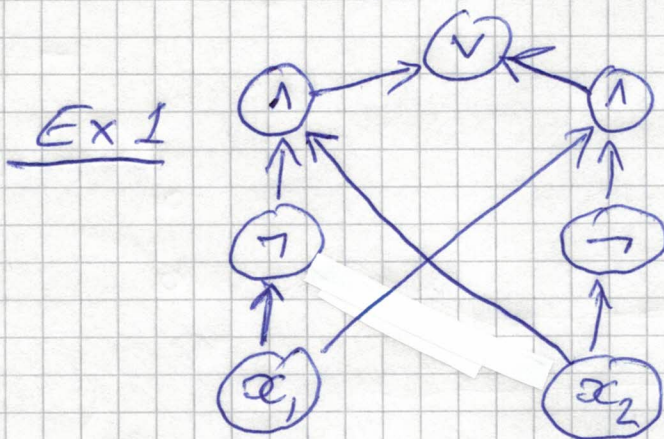
Labelled by

- o $\wedge$ (AND) fan-in 2
- o $\vee$ (OR) fan-in 2
- o $\neg$ (NOT) fan-in 1

C_n computes Boolean function

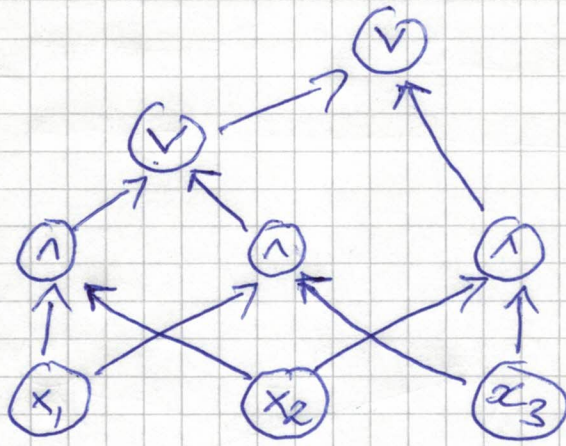
$f_n : \{0, 1\}^n \rightarrow \{0, 1\}$ as so:

- source vertices compute values of their variables
- gates compute Boolean function applied to incoming edges / wires
- output of C_n = value computed by sink



$x_1 \text{ XOR } x_2$

" x_1 and x_2 take opposite values"

Ex 2

$$\text{MAJ}(x_1, x_2, x_3)$$

"majority of x_1, x_2 ,
and x_3 are true"

Size of circuit = # nodes

Easy to prove: Any function $f: \{0,1\}^n \rightarrow \{0,1\}$
can be computed by a circuit of
size $O(n 2^n)$

Can do better: $O(2^n/n)$ always
possible

CIRCUIT COMPLEXITY: studies size of
circuits for Boolean functions.
Prove explicit lower bounds for
concrete functions

Most functions require size $\Omega(2^n/n)$
Cannot prove ^{even} superlinear lower
bounds except for restricted classes
of circuits

For $x, y \in \{0, 1\}^n$ write

$x \leq y$ if for all $i \in [n]$ $x_i \leq y_i$

Boolean function $f: \{0, 1\}^n \rightarrow \{0, 1\}$ is MONOTONE if $x \leq y \Rightarrow f(x) \leq f(y)$

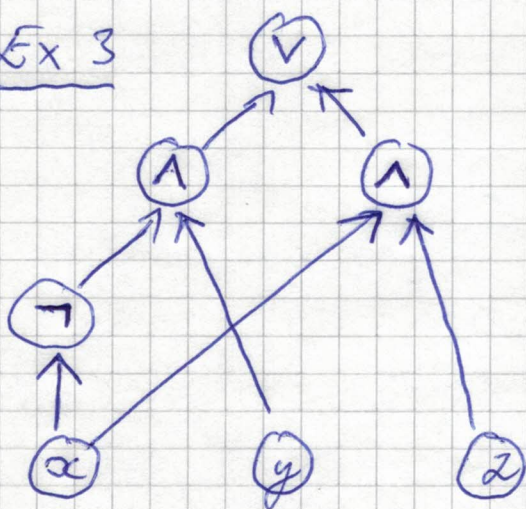
A Boolean circuit is monotone if it has only $\wedge$ - and $\vee$ -gates but no $\neg$ -gates

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

The circuit in Ex 1 is not monotone

The circuit in Ex 2 is monotone

Ex 3



SELECTOR function

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

sel is not monotone

INTERPOLATION AND CLIQUE-COLOURING FORMULAS PCI VII

Interpolation method introduced by
Krajíček 1994

Used by Pudlák in 1997 to prove lower bounds on proof length for clique-colouring formulas in the CUTTING PLANES proof system (much stronger proof system).

To make our lives a little bit easier, we will do a version of Pudlák's result for resolution, which is a weaker proof system

Clique-colouring formulas
Parameters

- n : # vertices in graph
- m : size of clique

Formula says:

"There exists a graph on n vertices which has an m -clique and is also $(m-1)$ -colourable"

Obviously absurd, so formula unsatisfiable
But resolution has a hard time understanding this...

Variables

$$\vec{P} = \{p_{ij} \mid 1 \leq i < j \leq n\}$$

p_{ij} = "there is an edge between vertices i & j "

$$\vec{q} = \{q_{k,i} \mid k \in [m], i \in [n]\}$$

$q_{k,i}$ = "vertex i is k th member of clique"

$$\vec{r} = \{r_{i,l} \mid i \in [n], l \in [m-1]\}$$

$r_{i,l}$ = "vertex i gets colour l "

Axiom clauses

$\bigvee_{i \in [n]} q_{k,i}$ "some vertex is k th clique member"

$\overline{q_{k,i}} \vee \overline{q_{k',i}}$ "clique vertices have unique member numbers" ($k \neq k'$)

$p_{ij} \vee \overline{q_{k,i}} \vee \overline{q_{k',j}}$ "clique members are connected by edges" ($i < j, k \neq k'$)

$\bigvee_{l \in [m-1]} r_{i,l}$ "every vertex gets a colour"

$\overline{p_{ij}} \vee \overline{r_{i,l}} \vee \overline{r_{j,l}}$ "neighbours have distinct colours"

Clique-colouring formula can be written as

$$A(\vec{p}, \vec{q}) \wedge B(\vec{p}, \vec{r}) \quad (*)$$

where the variable sets $\vec{p}, \vec{q}, \vec{r}$ are disjoint.

Given partial assignment, or restriction, g to variables $\text{Vars}(F)$ of CNF formula F , we write $F|_g$ for the restricted formula where variables $x \in \text{Dom}(g)$ are replaced by values $g(x)$ and formula is simplified by removing

- satisfied clauses
- falsified literals

Ex

$$F = (x \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{x} \vee y) \wedge (\bar{x} \vee z) \wedge (\bar{y} \vee \bar{z})$$

$$g = \{y \mapsto 1\}$$

$$F|_g = x \wedge (\bar{x} \vee z) \wedge \bar{z}$$

Suppose we have unsatisfiable formula

$$F = A(\vec{p}, \vec{q}) \wedge B(\vec{p}, \vec{r}) \quad \text{as in } (*)$$

and restriction g assigning all of $\vec{p}$

Then $F|_g = A(\vec{p}, \vec{q})|_g \wedge B(\vec{p}, \vec{r})|_g \quad \boxed{\text{PCI } \bar{x}}$

splits into two formulas on disjoint sets of variables.

Write

$$A(\vec{p}, \vec{q})|_g = A(g, \vec{q})$$

$$B(\vec{p}, \vec{r})|_g = B(g, \vec{r})$$

- for notational convenience

For any such g , either $A(g, \vec{q})$

- or $B(g, \vec{r})$ (or both) must be unsatisfiable.

A Boolean circuit $I(\vec{p})$ is an INTERPOLANT for $A(\vec{p}, \vec{q}) \wedge B(\vec{p}, \vec{r})$ if

$$I(g) = 0 \Rightarrow A(g, \vec{q}) \text{ unsatisfiable}$$

$$I(g) = 1 \Rightarrow B(g, \vec{r}) \text{ unsatisfiable}$$

- Such interpolants always exist (and are not necessarily unique)

We are going to prove

If $A(\vec{p}, \vec{q}) \wedge B(\vec{p}, \vec{r})$ has a short resolution refutation, then it also has a small interpolant.

But suppose interpolant computes function for which we have a circuit lower bound — then this shows that there cannot exist any short resolution refutations

PCI XI

PROOF STRATEGY

- ① Start with formula $F = A(\vec{p}, \vec{q}) \wedge B(\vec{p}, \vec{r})$
- ② Assume towards contradiction that F has short resolution refutation
- ③ Deduce that there exist small interpolants
- ④ But argue now that interpolant computes something for which we have circuit lower bound
- ⑤ Contradiction! Hence refutation cannot be short

Proof systems for which this proof strategy works are said to have **FEASIBLE INTERPOLATION**. Resolution (and cutting planes) has feasible interpolation.

Use this to show that clique-colouring formulas are hard for resolution

PROBLEM: We don't have good general circuit lower bounds!

SOLUTION: There is a monotone version of interpolation, and we are lucky enough that this will work

THEOREM 1 [Razborov '85, Alon-Boppana '87]

Let undirected graph G on n vertices be represented by $\binom{n}{2}$ edge indicator bits.

Then for $m = \Theta(\sqrt[4]{n})$ there is no monotone circuit of size $2^{o(\sqrt{m})}$ that can distinguish between

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

But an interpolating formula distinguishes precisely these two cases!

Define ternary SELECTOR function by

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

This function is not monotone (why?)

We will build interpolating circuit using "gates" $\{\wedge, \vee, \text{sel}\}$.

Later, we will argue that we can replace sel by just $\wedge$ - and $\vee$ -gates to get monotone interpolating circuit.

THEOREM 2 [Pudlak '97]

MI II

Suppose $A(\vec{p}, \vec{q}) \wedge B(\vec{p}, \vec{r})$ is an unsatisfiable CNF formula over disjoint sets of variables $\vec{p}, \vec{q}, \vec{r}$.

Let $\pi: A \wedge B \vdash \perp$ be a resolution refutation in length L . Then the following holds:

- ① There is an INTERPOLATING CIRCUIT $I(p)$ of size $O(L)$ such that
 - (a) $I(p) = 0 \Rightarrow A(p, \vec{q})$ unsatisfiable
 - (b) $I(p) = 1 \Rightarrow B(p, \vec{r})$ unsatisfiable

- ② From π one can construct resolution refutation

- (a) $\pi_A: A(p, \vec{q}) \vdash \perp$ if $I(p) = 0$
 - (b) $\pi_B: B(p, \vec{r}) \vdash \perp$ if $I(p) = 1$
- in both cases of length $\leq L$

- ③ If $\vec{p}$ -variables occur only positively in $A(\vec{p}, \vec{q})$ or only negatively in $B(\vec{q}, \vec{r})$, then the circuit $I(p)$ can be made monotone

COROLLARY 3

Clique-colouring formulas with $m = \Theta(\sqrt[4]{n})$ require resolution refutations of length $\exp(\Omega(\sqrt[8]{n}))$.

Build circuits with $\wedge$, $\vee$, and ternary sel-gates for simplicity

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

sel-gates are not monotone, but we'll discuss how to fix this later.

PROOF PLAN

- First do part (2)
- Then use (2) to get (1)
- Finally, talk about the monotonicity in (3)

Notational convention: If φ satisfies clause C , write $C|_{\varphi} = 1$ and consider I to be a "trivial clause"

Clause types

g -clause: Clause C over variables $\vec{z}$ derivable from $A(\varphi, \vec{z})$

r -clause: Clause C over variables $\vec{r}$ derivable from $B(\varphi, \vec{r})$

Clauses I can be both g -clause and r -clause

Inductive proof of part (2)

From $\pi = (C_1, C_2, \dots, C_L)$ construct sequence of clauses $\tilde{\pi} = (\tilde{C}_1, \tilde{C}_2, \dots, \tilde{C}_L)$ which contain candidate derivations π_A and π_B

Inductive hypotheses

(P1) $\tilde{C}_i$ is a g-clause or an r-clause

Write $\text{type}(\tilde{C}_i) = g/r$

(P2) $\tilde{C}_i = 1$ only if $C_i|_g = 1$, and if $\tilde{C}_i \neq 1$ it holds that $\tilde{C}_i \subseteq C_i$

(P3) If $\tilde{C}_i = 1$, then there is associated axiom clause E_i such that

- E_i satisfied by $f(a) = 1$ for $a \in C_i \cap E_i$
- $E_i \in A(\vec{p}, \vec{q})$ if $\text{type}(\tilde{C}_i) = g$
- $E_i \in B(\vec{p}, \vec{r})$ if $\text{type}(\tilde{C}_i) = r$

Think of E_i as "justification" / "excuse" why we chose $\tilde{C}_i = 1$.

This book-keeping important only for monotonicity in part (3), so we can mostly ignore E_i until then.

We now need to select clauses $\tilde{C}_i$ and types g/r inductively for all $C_i \in \pi$.

Base case $C_i \in A(\vec{p}, \vec{q}) \cup B(\vec{p}, \vec{r})$

MI V

Set $\tilde{C}_i = C_i \setminus p$

$E_i = C_i$ if justification needed

$$\text{type}(\tilde{C}_i) = \begin{cases} q & \text{if } C_i \in A(\vec{p}, \vec{q}) \\ r & \text{if } C_i \in B(\vec{p}, \vec{r}) \end{cases}$$

Properties (P1) - (P3) clearly hold

Inductive step

$C_i = C \vee D$ derived by resolution rule

$$\frac{C \vee x \quad D \vee \bar{x}}{C \vee D}$$

from

$$C_j = C \vee x$$

$$j < i$$

$$C_k = D \vee \bar{x}$$

$$k < i$$

By induction, have already constructed $\tilde{C}_j, \tilde{C}_k$

Case analysis for $x \in \vec{p} \cup \vec{q} \cup \vec{r}$

Case 1: $x \in \vec{p}$

$$\begin{aligned} \text{If } p(x) = 0 \text{ set } \tilde{C}_i &:= \tilde{C}_j \\ \text{type}(\tilde{C}_i) &:= \text{type}(\tilde{C}_j) \\ E_i &:= E_j \text{ (if needed)} \end{aligned}$$

(P1) clearly OK

If $\tilde{C}_i \neq 1$ then

$$\begin{aligned} \tilde{C}_i = \tilde{C}_j &\subseteq C_j \setminus \{x\} \\ &\subseteq C \vee D \\ &= C_i \end{aligned}$$

If $\tilde{C}_i = 1$, then $C \upharpoonright_g = 1$ since $g(x) = 0$,
 so $C_i \upharpoonright_g = (C \vee D) \upharpoonright_g = 1$ and $E_i = E_j$
 works as justification axiom.

(P2) and (P3) are OK

If instead $g(x) = 1$, set $\tilde{C}_i := \tilde{C}_k$
 $\text{type}(\tilde{C}_i) := \text{type}(\tilde{C}_k)$
 $E_i := E_k$

Argument exactly the same as for $g(x) = 0$

Case 2: $x \in \vec{q}$

Divide analysis into subcases depending
 on types of $\tilde{C}_j$ and $\tilde{C}_k$

(a) If one of $\tilde{C}_j$ and $\tilde{C}_k$ is r -clause, then
 we have already gotten rid of x in this clause
 so set $\tilde{C}_i$ to this clause (choose arbitrarily
 if both are r -clauses).

Set $\text{type}(\tilde{C}_i) = r$ (P1) clearly OK
 Copy justification clause if needed

Observe:

- $\tilde{C}_i$ doesn't contain any $\vec{q}$ -variables (by IH)
- $x \in \vec{q}$ only variable that disappears in
 resolution step.

Therefore (P2) and (P3) OK.

Otherwise
 (b) If either $\tilde{C}_j$ or $\tilde{C}_k$ g -clause not containing literal x (e.g., if one of them = 1), then let $\tilde{C}_i$ be this clause (pick arbitrarily if both qualify) and set $\text{type}(\tilde{C}_i) = g$.
 Copy justification clause if needed.

Note that since no variable in β disappears in the resolution step, the justification clause is still OK for C_i , and $\tilde{C}_i = 1$ only if $C_i|_{\beta} = 1$.

(c) Otherwise, we have two non-trivial g -clauses

$$\tilde{C}_j = \tilde{C}_j' \vee x$$

$$\tilde{C}_k = \tilde{C}_k' \vee \bar{x}$$

Set $\tilde{C}_i$ to be the resolvent $\tilde{C}_j' \vee \tilde{C}_k'$ and $\text{type}(\tilde{C}_i) = g$.

(P1): Resolvent of two g -clauses is a g -clause

$$\begin{aligned} \text{(P2): } \tilde{C}_i &\subseteq (\tilde{C}_j \vee \tilde{C}_k) \setminus \{x, \bar{x}\} \\ &\subseteq C_j \vee C_k \\ &= C_i \end{aligned}$$

(P3) Vacuously OK, since $\tilde{C}_i|_{\beta} \neq 1$

Case 3 ($x \in \vec{r}$)

MI VII

Analogous to Case 2

If one of $\tilde{C}_i$ and $\tilde{C}_k$ is g -clause, then copy that clause.

Otherwise copy or construct r -clause.

Details left as an exercise

Part (2) now follows by the induction principle.

Final clause $C_L = 1$ gets classified as g -clause or r -clause

$\tilde{C}_L \neq 1$ since $C_L|_g = 1|_g = 1 \neq 1$,

so $\tilde{C}_L \leq C_L$ meaning $\tilde{C}_L = 1$

This means:

If $\text{type}(\tilde{C}_L) = g$, then $A(p, \vec{g})$ unsatisfiable

If $\text{type}(\tilde{C}_L) = r$, then $B(p, \vec{r})$ unsatisfiable.

Proof of part (1)

Go over $\Pi = (C_1, C_2, \dots, C_L)$

Look at construction of $\tilde{\Pi} = (\tilde{C}_1, \tilde{C}_2, \dots, \tilde{C}_L)$

For each C_i , construct subcircuit with output gate v_i computing

$$\text{type}(\tilde{C}_i) = \begin{cases} 0 & \text{if } \tilde{C}_i \text{ } g\text{-clause} \\ 1 & \text{if } \tilde{C}_i \text{ } r\text{-clause} \end{cases}$$

Then circuit with output gate v_L is interpolant

Add constant #gates per C_i = same size as resolution proof

Just inspect proof of part ②

M/IX

Base case

$$C_i \in A(\vec{p}, \vec{q}) \Rightarrow v_i \text{ constant } 0$$

$$C_i \in B(\vec{p}, \vec{r}) \Rightarrow v_i \text{ constant } 1$$

Induction step

$$\frac{C \vee x \quad D \vee \bar{x}}{C \vee D}$$

Again case analysis over variable $x \in \vec{p} \cup \vec{q} \cup \vec{r}$
resolved over

Case 1: $x \in \vec{p}$

$$\begin{aligned} \text{type}(\tilde{C}_i) &= \text{sel}(x, \text{type}(\tilde{C}_j), \text{type}(\tilde{C}_k)) \\ &= \text{sel}(x, v_j, v_k) \end{aligned}$$

Case 2: $x \in \vec{q}$

$$\text{type}(\tilde{C}_i) = \begin{cases} 1 & \text{if one of } \tilde{C}_j, \tilde{C}_k \text{ has type } 1 \\ 0 & \text{otherwise} \end{cases}$$

Or, in other words:

$$\begin{aligned} \text{type}(\tilde{C}_i) &= \text{type}(\tilde{C}_j) \vee \text{type}(\tilde{C}_k) \\ &= v_j \vee v_k \end{aligned}$$

Case 3: $x \in \vec{r}$

$$\begin{aligned} \text{type}(\tilde{C}_i) &= \text{type}(\tilde{C}_j) \wedge \text{type}(\tilde{C}_k) \\ &= v_j \wedge v_k \end{aligned}$$

Details left as exercise.

Clearly, constant #gates per step in resolution proof
 $\text{type}(\tilde{C}_k) = \text{type}(1)$ computed by output gate v_k
Yields correct interpolating circuit of size $O(L)$

Proof of part ③

III x

Selector gates are non-monotone.

Need to get rid of them.

Will use "justification axioms" to do so.

Assume $\vec{p}$ -variables appear only positively in $A(\vec{p}, \vec{q})$. (If $\vec{p}$ -variables appear only negatively in $B(\vec{p}, \vec{q})$, then we can do an analogous argument, or just flip all $\vec{p}$ -variables to their negation and change roles of A and B.)

Replace all gates

$$\text{sel}(x, u, v) = (x \vee u) \wedge (\bar{x} \vee v)$$

by

$$(x \vee u) \wedge v$$

The resulting circuit is monotone
But computations in sel-gates have changed

x	u	v	sel(x, u, v)	$(x \vee u) \wedge v$
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	1	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

So we will get an error when sel-gates have inputs $(0, 1, 0)$
When does this happen?

11/81

$$\frac{C_j = C_v x \quad C_k = D v \bar{x}}{C_i = C_v D} \quad x \in \vec{p}$$

and $g(x) = 0$

$$\begin{aligned} \text{type}(\tilde{C}_j) &= r \\ \text{type}(\tilde{C}_k) &= g \end{aligned}$$

What can go wrong?

According to our proof, we should set

$$\begin{aligned} \tilde{C}_i &:= \tilde{C}_j \\ \text{type}(\tilde{C}_i) &:= r \end{aligned}$$

But our modified circuit says

$$\begin{aligned} \tilde{C}_i &:= \tilde{C}_k \\ \text{type}(\tilde{C}_i) &:= g \end{aligned}$$

What can go wrong? Nothing, if $\tilde{C}_k \neq 1$.

But suppose we have

$$C_i \upharpoonright_{\mathcal{P}} = (C_v D) \upharpoonright_{\mathcal{P}} \neq 1 \quad (+)$$

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

$$C_k \upharpoonright_{\mathcal{P}} = (D v \bar{x}) \upharpoonright_{\mathcal{P}} = 1 \quad (\neq)$$

Then we are violating (P2).

NOT ALLOWED!

$$\begin{aligned} \tilde{C}_i &= 1 \\ C_i \upharpoonright_{\mathcal{P}} &\neq 1 \end{aligned}$$

Suppose $\tilde{C}_k = 1$ for type $(\tilde{C}_k) = g$.

III. XII

Then $\exists \bar{E}_k \in A(\vec{p}, \vec{g})$ satisfied by $g(a) = 1$ for literal $a \in \bar{E}_k \cap C_k$

In view of $(+)$ and $(\neq)$, the only possibility is $a = \bar{x}$, because all other literals in C_k are also in C_i .

But clauses in $A(\vec{p}, \vec{g})$ only contain $\vec{p}$ -variables positively. So this can never happen, and our simplified monotone circuit will compute the same value as the circuit with non-monotone sub-gates.

This concludes the proof of Thm 2. 

Corollary 3 now follows from combining Thm 2 with the monotone circuit lower bound in Thm 1.