PROOF COMPLEXITY AS A COMPUTATIONAL LENS
LECTURE 4

Recap

- Resolution proof system
- lower bounds for PHP and Tseitin formulas (bundled using 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

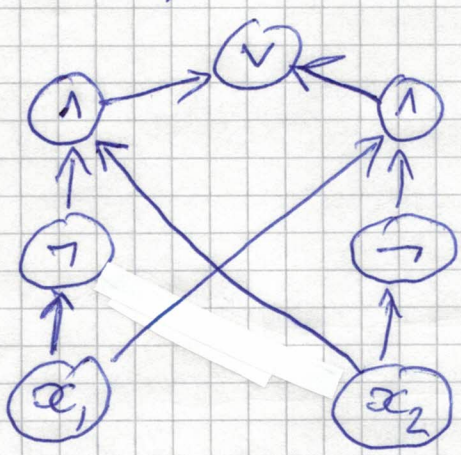
Directed acyclic graph (DAG) C_n
n sources (nodes without incoming edges)
unique sink (without outgoing edges)
Sources labelled by variables $x_1, \ldots, x_n$
Non-source vertices have in-degree / fan-in 1 or 2 and are called gates
Labelled by

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

C_n computes Boolean function $f_n : \{0,1\}^n \to \{0,1\}$ 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

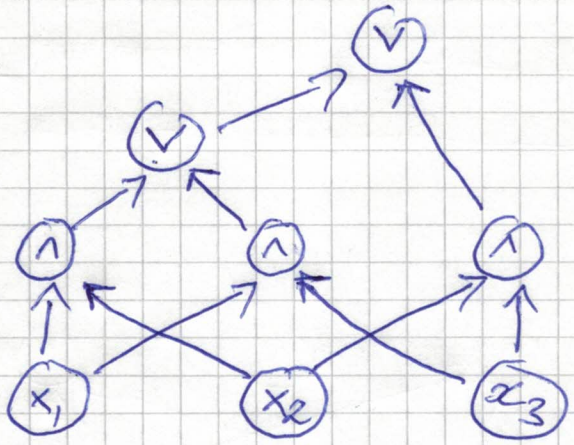
Ex 1



x_1 XOR x_2

"x_1 and x_2 take opposite values"

Ex 2



$MAJ(x_1, x_2, x_3)$

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

Size of circuit = # nodes

Easy to prove: Any function $f: \{0,1\}^n \to \{0,1\}$ 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 $i \in [n]$ $x_i \leq y_i$

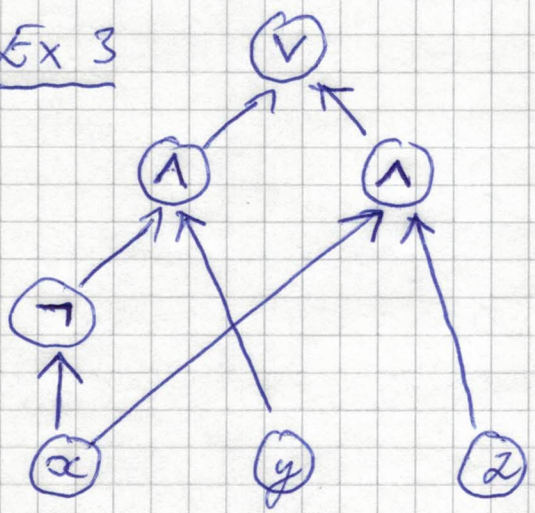
Boolean function $f: \{0,1\}^n \to \{0,1\}$ is **MONOTONE** if $x \leq y \Rightarrow f(x) \leq f(y)$

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

FACT A Boolean function $f: \{0,1\}^n \to \{0,1\}$ 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

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

sel is not monotone

INTERPOLATION AND CLIQUE-COLOURING FORMULAS

PC / 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

$\vec{p} = \{p_{i,j} \mid 1 \le i < j \le n\}$

$p_{i,j}$ = "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 kth member of clique"

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

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

Axiom clauses

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

$\bar{q}_{k,i} \vee \bar{q}_{k',i}$ "clique vertices have unique member numbers" $(k \neq k')$

$p_{i,j} \vee \bar{q}_{k,i} \vee \bar{q}_{k',j}$ "clique members are connected by edges" $(i<j,\ k \neq k')$

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

$\bar{p}_{i,j} \vee \bar{r}_{i,\ell} \vee \bar{r}_{j,\ell}$ "neighbours have distinct colours"

Clique-colouring formula can be written as

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

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

Given partial assignment, or restriction, ρ to variables $Vars(F)$ of CNF formula F, we write $F\restriction_\rho$ for the restricted formula where variables $x \in Dom(\rho)$ are replaced by values $\rho(x)$ and formula is simplified by removing

- satisfied clauses
- falsified literals

Ex

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

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

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

Suppose we have unsatisfiable formula $F = A(\vec{p},\vec{q}) \wedge B(\vec{p},\vec{r})$ as in $(*)$ and restriction ρ assigning all of $\vec{p}$

Then $F\restriction_\rho = A(\vec{p},\vec{q})\restriction_\rho \wedge B(\vec{p},\vec{r})\restriction_\rho$ splits into two formulas on disjoint sets of variables.

PCI ☒

Write

$$A(\vec{p},\vec{q})\restriction_\rho = A(\rho,\vec{q})$$

$$B(\vec{p},\vec{r})\restriction_\rho = B(\rho,\vec{r})$$

- for notational convenience
- For any such ρ, either $A(\rho,\vec{q})$
- or $B(\rho,\vec{r})$ (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

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

- $I(\rho) = 1 \Rightarrow B(\rho,\vec{r})$ 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

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

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

But an interpolant for the clique-colouring formula distinguishes precisely these two cases ☹

What we need to do

① Show how to build interpolating circuit efficiently from resolution refutation π: $A(\vec{p}, \vec{q}) \wedge B(\vec{p}, \vec{r}) \vdash \bot$

② Show that this circuit can be made monotone if formula $F = A(\vec{p}, \vec{q}) \wedge B(\vec{p}, \vec{r})$ has the right structure

MI II

THEOREM 2 [Pudlák '97]

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 $\pi : A \wedge B \vdash \bot$ 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) $\pi_A : A(p,\vec{q}) \vdash \bot$ if $I(p)=0$
   (b) $\pi_B : B(p,\vec{r}) \vdash \bot$ 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 ρ satisfies clause C, write $C \restriction \rho = 1$ and consider 1 to be a "trivial clause"

Clause types

q-clause: Clause C over variables $\vec{q}$ derivable from $A(\rho, \vec{q})$

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

Clauses 1 can be both q-clause and r-clause

Inductive proof of part (2)

M ⊨ N

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

Inductive hypotheses

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

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

(P2) $\tilde{C}_i = 1$ only if $C_i|_S = 1$, 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 $S(a) = 1$ for $a \in C_i \cap E_i$
- $E_i \in A(\vec{p}, \vec{q})$ if $\text{type}(\tilde{C}_i) = q$
- $E_i \in B(\vec{p}, \vec{r})$ 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 q/r inductively for all $C_i \in \pi$.

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

Set $\tilde{C}_i = C_i \wedge \beta$

$\mathcal{E}_i = C_i$ if justification needed

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

Properties (P1) - (P3) clearly hold

Inductive step

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

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

from

$C_j = C \vee x$ $\quad j < i$

$C_k = D \vee \bar{x}$ $\quad k < i$

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: $x \in \vec{p}$

If $\beta(x) = 0$ set $\tilde{C}_i := \tilde{C}_j$

$type(\tilde{C}_i) := type(\tilde{C}_j)$

$\mathcal{E}_i := \mathcal{E}_j$ (if needed)

(P1) clearly OK

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

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

If $\tilde{C}_i = \underline{1}$, then $C \wedge \rho = \underline{1}$ since $\rho(x) = 0$,
so $C_i \wedge \rho = (C \vee D) \wedge \rho = \underline{1}$ and $\mathcal{E}_i = \mathcal{E}_j$
works as justification axiom.
(P2) and (P3) are OK

If instead $\rho(x) = \underline{1}$, set $\tilde{C}_i := \tilde{C}_k$

$\text{type}(\tilde{C}_i) := \text{type}(\tilde{C}_k)$

$\mathcal{E}_i := \mathcal{E}_k$

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

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 τ-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 τ-clauses.

Set $\text{type}(\tilde{C}_i) = \tau$ (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.

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

Note that since no variable in $\vec{p}$ disappears in the resolution step, the justification clause is still OK for C_i, and $\tilde{C}_i = 1$ only if $C_i \wedge q = 1$

(c) Otherwise, we have two non-trivial q-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 type$(\tilde{C}_i) = q$

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

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

(P3) Vacuously OK, since $\tilde{C}_i \wedge q \neq 1$

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

Analogous to Case 2
If one of $\tilde{C}_i$ and $\tilde{C}_k$ is q-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 $C_L = \bot$ gets classified as q-clause or r-clause

$\tilde{C}_L \neq 1$ since $C_L|_\sigma = \bot|_\sigma = \bot \neq 1$,
so $\tilde{C}_L \leq C_L$ meaning $\tilde{C}_L = \bot$

This means:
If $type(\tilde{C}_L) = q$, then $A(\sigma, \vec{q})$ unsatisfiable
If $type(\tilde{C}_L) = r$, then $B(\sigma, \vec{r})$ unsatisfiable.

Proof of part ①

Go over $\pi = (C_1, C_2, \ldots, C_L)$
Look at construction of $\tilde{\pi} = (\tilde{C}_1, \tilde{C}_2, \ldots, \tilde{C}_L)$

For each C_i, construct subcircuit with output gate v_i computing

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

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)

Base case

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

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

Induction step $\quad \dfrac{C \vee x \quad D \vee \bar{x}}{C \vee D}$

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

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

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

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

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

Or, in other words:

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

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

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

Details left as exercise.

Clearly, constant # gates per step in resolution proof

$type(\tilde{C}_\perp) = type(\perp)$ computed by output gate $v_\perp$

Yields correct interpolating circuit of size $O(L)$

Proof of part (3)

MI $\overline{x}$

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 $B(\vec{p}, \vec{r})$, 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

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

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?

$$\frac{C_j = C \vee x \qquad C_k = D \vee \bar{x}}{C_i = C \vee D} \qquad x \in \vec{p}$$

and $\varphi(x) = 0$

$$type(\tilde{C}_j) = r$$
$$type(\tilde{C}_k) = g$$

What can go wrong?

According to our proof, we should set

$$\tilde{C}_i := \tilde{C}_j$$
$$type(\tilde{C}_i) := r$$

But our modified circuit says

$$\tilde{C}_i := \tilde{C}_k$$
$$type(\tilde{C}_i) := g$$

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

But suppose we have

$$C_i \wedge \varphi = (C \vee D) \wedge \varphi \neq 1 \qquad (\dagger)$$

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

$$C_k \wedge \varphi = (D \vee \bar{x}) \wedge \varphi = 1 \qquad (\dagger)$$

Then we are violating (P2).

NOT ALLOWED!

$$\tilde{C}_i = 1$$
$$C_i \wedge \varphi \neq 1$$

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

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

In view of $(\dagger)$ and $(\ddagger)$, 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.