LECTURE 27: AUTOMATABILITY

If we want to use a proof system P to solve computational problems, what properties do we want P to have?

① P should be **POWERFUL**: Given unsatisfiable formula F, want small refutations $\pi : F \vdash \bot$

② P should admit **EFFICIENT PROOF SEARCH**: Given unsatisfiable formula F, should be possible to find refutation $\pi : F \vdash \bot$ quickly.

How to measure efficiency?

- If shortest proof has exponential length, then need exponential time
  So require running time $\text{poly}(S_P(F \vdash \bot))$
- Edge case: Consider formula
  $F = \text{GiganticMess} \wedge x \wedge \neg x$
  Constant-size proof, but we cannot find it without parsing formula

DEFINITION (AUTOMATABILITY) adapted from [B.P.R '00]

A proof system P is **AUTOMATABLE** if there is an algorithm that when given unsatisfiable CNF formula F outputs a P-refutation of F in time $\text{poly}(S(F) + S_P(F \vdash \bot))$.

Can also study more generous notions of automatability in quasi-polynomial time, et cetera

Are the proof systems we have studied automatable

n = # variables

WIDTH- / DEGREE- AUTOMATABILITY

If F has a resolution, Nullstellensatz, or polynomial calculus refutation in width/degree d, then such a refutation can be found in time $n^{O(d)}$

But there are formulas with refutation width/degree $\Omega(\sqrt{n})$ for resolution and PC and degree $\Omega(n/\log n)$ for NS but polynomial refutation size (in fact, linear in formula size)

[Alekhnovich-Razborov '08]
Resolution is not automatable unless parameterized complexity hierarchy collapses

[Galesi-Lauria '10]
Polynomial calculus is not automatable under same assumption

A proof system is WEAKLY AUTOMATABLE if it is polynomially simulated by an automatable proof system

For any proof system that is closed under restrictions, weak automatability implies feasible interpolation [BPR '00]

For strong enough proof systems
(that have short proofs of their own soundness)
weak automatability is equivalent to
feasible interpolation [Pudlák '03]
(Frege and extended Frege)

Under cryptographic assumptions
- bounded-depth Frege
- Frege
- extended Frege

do not have feasible interpolation,
and so are not weakly automatable
[KP '98, BPR '00, BDGMP '04]

Breakthrough by Atserias & Müller [AM20]
Resolution is not automatable
unless NP ⊆ P

(Optimal assumptions: if NP ⊆ P, then
resolution is automatable)

Has led to other non-automatability
results for
- Nullstellensatz & polynomial calculus
  [dRGNPRS '21]
- cutting planes [GKMP '20]
- k-DNF resolution [Garlik '20]
- tree-like resolution (under ETH)
  [de Rezende '21]

OPEN FOR: Sherali-Adams & sum-of-squares

What we would like to cover today

THEOREM 1 [dR GNPRS '21, building on AM '20]
There is a poly-time algorithm A that
- when given 3-CNF formula F over n variables
- outputs CNF formula $A(F)$ such that

for P = resolution, polynomial calculus, or Nullstellensatz:
- if F is satisfiable, then $A(F)$ has P-refutation of size $n^{O(1)}$
- if F is unsatisfiable, then $A(F)$ requires P-refutations of size at least $\exp(n^{\Omega(1)})$

Define
QP: problems solvable in time $\exp(\log^{O(1)} n)$
SUBEXP: -"- $\exp(n^{o(1)})$

COROLLARY 2
For P = resolution, polynomial calculus, or Nullstellensatz:
(a) P is not automatable in polynomial time unless $NP \subseteq P$
(b) P is not automatable in quasi-polynomial time unless $NP \subseteq QP$
(c) P is not automatable in subexponential time unless $NP \subseteq SUBEXP$

Proof sketch for corollary

Suppose P is automatable.
Use proof search algorithm S to solve 3-SAT
Given 3-CNF formula F
Compute CNF formula $A(F)$
Run S on $A(F)$ with polynomial time-out
Case analysis:

(i) F satisfiable:
Then $\exists$ short P-refutation of $A(F)$
S will find this refutation

(ii) F unsatisfiable
Then there is no short P-refutation, so S will time out.

This decides whether $F \in$ 3-SAT in polynomial time ∎

Remark
Algebraic results hold over any field and with or without dual variables.

Observation
When F satisfiable, short refutations of $A(F)$ must require large width/degree
Otherwise width-/degree-bounded search would find them efficiently.

Let us give a more detailed version
(but still deferring details until later)

Given 3-CNF formula F over n variables
A computes CNF formula

$$\underline{Ref(F,s)} = \boxed{\text{"}F \text{ has resolution refutation in length } s\text{"}}$$

We will fix $s = n^6$

except free weakening after every resolution step

$Ref(F,s)$ describes
- clauses $C_1, C_2, \ldots, C_s$
  (with variables indicating literals in C_i)
- how these clauses constitute a refutation

All variables describing a clause form a BLOCK

BLOCK-WIDTH $\quad bw(D) = \#$ blocks mentioned

LEMMA 3 [AM20]
$Ref(F,s)$ is a block-width-$O(1)$ formula such that
(i) If F is satisfiable, then there is a resolution refutation $\pi : Ref(F,s) \vdash \bot$ in length $L(\pi) = n^{O(1)}$ and block-width $bw(\pi) = O(1)$
(ii) If F is unsatisfiable, then $bw(Ref(F,s) \vdash \bot) = \tilde{\Omega}(n^4)$

Proof intuition

(i) If F satisfiable, easy to prove no refutation exists. Consider Prosecutor strategy. Fix assignment α satisfying F. Check that α violates $C_S = \bot$. Walk backwards in proof along falsified premises — blocks-width 3 — along clauses falsified by fixed assignment α. But α satisfies all axioms, so somewhere reach contradiction.

(ii) Two cases:

(a) F in fact has proof in length $\leq S$. Then $\mathrm{Ref}(F, S)$ satisfiable, and lower bound holds vacuously

(b) F is unsatisfiable, but $\ell(F \vdash \bot) > S$

WORK NEEDED...

LEMMA 4 [dRGNPRS '21]

If F unsatisfiable CNF formula over n variables then

$$bW(\mathrm{Ref}(F, s) \vdash \bot) = \tilde{\Omega}\left(W(\lceil PHP_{s/n} \vdash \bot) / n\right)$$

$\lceil PHP_m$ = "$\exists$ efficiently invertible injection from $2m$ pigeons to m holes"

Exist functions $f : [2m] \to [m]$, $g : [m] \to [2m]$

s.t. $f(i) = j \implies g(j) = i$

Then use:

$$W(\neg PHP_m \vdash \bot) = \Omega(m)$$

[Pudlák - Thapen '19]

How to get from block-width lower bounds to length lower bounds?

(1) Lifting / relativization [AM '20]

(2) Length - width lower bound [BW '01]

$$L(F \vdash \bot) = \exp\left(\frac{(W(F \vdash \bot) - W(F))^2}{\#\text{vars in } F}\right)$$

We can use (2) if

- use binary and not unary encodings (as in [AM20]) for $Ref(F, s)$
- pick s large enough (like $s = n^6$)

How to generalize this to Nullstellensatz and polynomial calculus?

PROBLEM: Not clear how Nullstellensatz can "walk backwards" in resolution proof.

FIX: Strengthen the formula to

$TreeRef(F, s)$ = "F has a tree-like resolution refutation in length s where weakening is only applied to axiom clauses"

Then prove versions of Lemmas 3 and 4 for (block-) degree.

ENCODING OF Ref(F, s)

F CNF formula with variables $x_1, \ldots, x_n$ and $m = \text{poly}(n)$ clauses

Variables of Ref(F, s) partitioned into $s = n^6$ blocks $B_1, B_2, \ldots, B_s$

Intuition: Each block B_i represents clause C_i in purported refutation

Variables for a block

- Literal indicators y_ℓ for $\ell \in \{x_i, \bar{x}_i \mid i \in [n]\}$ indicating literals in clause
- Axiom pointer $\lceil \log m \rceil$ bits encoding axiom index $j \in [m]$
- Premise pointers $2 \times \lceil \log s \rceil$ bits encoding indices of premises for resolution rule
- Resolved variable pointer $\lceil \log n \rceil$ bits
- Block type $\tau = (\tau_1, \tau_2) \in \{0,1\}^2$ where
  - $(0,0)$ = axiom clause
  - $(0,1)$ = derived clause
  - $(1,*)$ = disabled block

Constraints in $Ref(F, s)$

Contradiction C_s

Last block B_s should be enabled
clause should contain no literals

Derived clause C_i $\quad \tau = (0, 1)$

Premise pointers j, j' should be $< i$
Blocks B_j, $B_{j'}$ should be enabled
For resolved variable pointer k
C_j should contain x_k
$C_{j'}$ should contain $\bar{x}_k$
All literals in $(C_j \vee C_{j'}) \setminus \{x_k, \bar{x}_k\}$ appear in C_i

Axiom clause C_i $\quad \tau = (0, 0)$

For axiom pointer j, every literal in jth axiom clause $A_j \in F$ should also appear in C_i

Disabled block $\quad \tau = (1, *)$

No constraint imposed on variables
(i.e., all clauses constraining a block contain the literal $\bar{\tau}_1$)

All of these constraints are over $O(\log n)$ variables

Any way of writing clausal constraints over these variables is fine (and we can derive different encodings from each other in length $\exp(O(\log n)) = \text{poly}(n)$)

Additional variables and constraints in TreeRef(F,s)

- Resolvent pointer "logs" variables pointing to unique resolvent C_p
- Tree-likeness axioms
  If C_i derived from C_j and $C_{j'}$ then resolvent pointers in B_j & $B_{j'}$ should be i
  Resolvent pointer for C_i should point to C_p, $p > i$, having C_i as premise (unless $C_i = \bot$ for $i = s$)
- No weakening
  For C_i derived from C_j & $C_{j'}$ require

$$\boxed{\begin{aligned} C_i &\subseteq C_j \\ C_i &\subseteq C_{j'} \end{aligned}}$$

This means that

$$C_j = C_i \vee x_k$$
$$C_{j'} = C_i \vee \overline{x_k}$$

Axiom blocks can still encode weakenings of axioms

RETRACTION WEAK PIGEONHOLE PRINCIPLE FORMULA A XII

$\neg PHP_m$

$2m$ pigeons, m holes

Variables encode

- **PIGEON MAP** $f : [2m] \to [m]$
  with variables $f_{i,k}$ $i \in [2m]$, $k \in \lceil \log m \rceil$
- **HOLE MAP** $g : [m] \to [2m]$
  with variables $g_{j,l}$ $j \in [m]$, $l \in \lceil \log 2m \rceil$

Axiom clauses of $\neg PHP_m$

$\forall i \in [2m] \; \forall j \in [m]$

$$f(i) = j \Rightarrow g(j) = i$$

$O(m \log m)$ variables

$O(m^2 \log m)$ clauses of width $O(\log m)$

Let T decision tree for Boolean function f

For leaf l in T, write $\rho(l)$ for partial assignment leading to l

For partial assignment ρ, let $C_{\neg\rho}$ be maximal clause falsified by ρ, i.e., $C_{\neg\rho}$ contains negations of all literals satisfied by ρ

If T decision tree of depth d for f, then

$$\bigwedge_{l \text{ leaf labelled } 0} C_{\neg\rho(l)}$$

is a d-CNF formula representing f

$\vec{x} = (x_1, \ldots, x_n)$ $\quad \vec{y} = (y_1, \ldots, y_m)$ $\quad$ As $\overrightarrow{x_{[n]}}$

DEF Let $F(\vec{x})$ and $G(\vec{y})$ be CNF formulas

A **DEPTH-d DECISION TREE REDUCTION** $F \leq_d^{dt} G$ is a function $f: \{0,1\}^n \to \{0,1\}^m$ such that

(1) Every output bit $f_i: \{0,1\}^n \to \{0,1\}$, $i \in [m]$ (which we think of as value given to y_i) is computed by a depth-d decision tree T_i

(2) For a clause $C(\vec{y}) \in G$, let T_C be the decision tree of depth $d \cdot W(C)$ that queries $\vec{x}$ for the value of the first y-variable, if the literal is false then for the second, et cetera, so that T_C computes $C \circ f$. Then for every clause D in the $d \cdot W(C)$-CNF representation of $C \circ f$ it should hold that D is a weakening of some axiom in F.

LEMMA 5

If $F \leq_d^{dt} G$, then $W(F \vdash \bot) \leq d \cdot W(G \vdash \bot) + O(1)$

Proof sketch

Use Prosecutor strategy for G.
When y_i queried, use T_i to issue $\leq d$ queries to $\vec{x}$ and remember answers
When y_i forgotten, forget queries by T_i
When Prosecutor reaches winning position for $C(\vec{y})$ a (weakening of) some axiom in F also falsified

DEFINITION

Suppose that variables $\vec{y}$ of G partitioned into blocks. Then a depth-d decision tree reduction is BLOCK-AWARE if for each block B there is a depth-d decision tree T_B that computes all values $f_B(\vec{x}) = (f_i(x) : i \in B) \in \{0,1\}^B$ simultaneously

LEMMA 6

If $F \leq^{dt}_{d} G$ via a block-aware reduction, then $W(F \vdash \bot) \leq d \cdot GW(G \vdash \bot) + O(1)$.

Proof sketch Just use the Prosecutor-Defendant game but for block-width instead of width ☐

(Part (ii) of Lemma 3 now follows from Lemma 4, and hence)

$W(\neg PHP^m_{m-1} \vdash \bot) = \Omega(m)$ [PT '19]

if we can prove following lemma

LEMMA 7

If F is an unsatisfiable CNF formula over n variables and $s = n^6$, then

$$\neg PHP^{n^5}_{n^5-1} \leq^{dt}_{\tilde{O}(n)} Ref(F,s)$$

via a block-aware reduction

$Ref(F,s)$ has $O(ns) = O(n^7)$ variables

Let $N = n^7$. Then Lemma 7 says that

$$W(Ref(F,s) \vdash \bot) \geq GW(Ref(F,s) \vdash \bot) \geq n^5 / \tilde{O}(n) = \tilde{\Omega}(n^4) \gg \sqrt{N}$$

Using that

$$\mathcal{L}(Ref(F,s)\vdash 1) = \exp\left(\frac{[W(Ref(F,s)\vdash 1) - W(Ref(F,s))]^2}{N}\right)$$

$$= \exp(\Omega(N^{\delta})) \quad \text{for some } \delta > 0$$

we get the lower bound in Thm 1.

Now let us prove Lemma 7

Consider the canonical tree-like refutation $\pi_T : F \vdash \bot$ that has 2^n leaves corresponding to all width-n clauses (which are all weakenings of axioms of F) and that for all transitions from depth $i+1$ to depth i resolves on x_i

Our lower bound strategy is to claim that $S = n^6$ clauses is enough to embed this refutation of length $2^{n+1} - 1$

Arrange the blocks in reverse order $B_S, B_{S-1}, \ldots, B_2, B_1$ into $n+1$ layers $L_0, L_1, \ldots, L_n$ where layer L_i contains $\min\{2^i, n^5\}$ blocks

Any blocks not needed will be defined as disabled

Say that a FULL-SIZE layer L_i has n^5 blocks

The blocks in a full-size layer L_i say that there are $2 \cdot n^5$ premise clauses at layer L_{i+1}, the pairwise resolvents of which are the clauses at layer L_i

Premise clause pointers = pigeons
Blocks at layer below = pigeonholes

Use $\lceil PHP_{n^5} \rceil$ to claim that this is perfectly possible

Hard-code non-full-size layers

Between full-size layer, any assignment to $Vars(\lceil PHP_{n^5} \rceil)$ defines partial mapping

$i \leftrightarrow j$ when $f(i) = j \wedge g(j) = i$

Use this mapping to define premise pointers

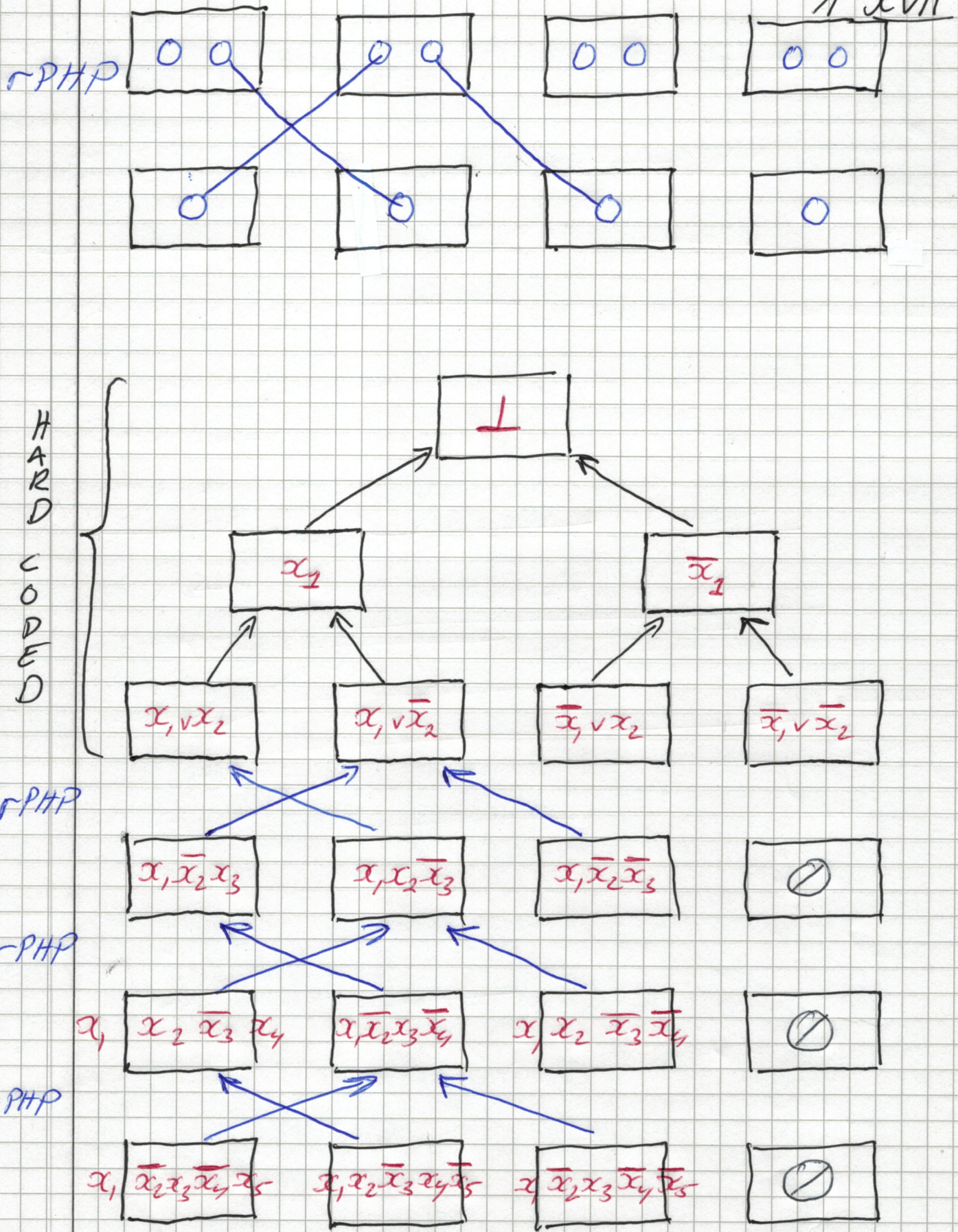
Define partial mapping $h : [2n^5] \to [n^5]$ by

$$h(i) = \begin{cases} f(i) & \text{if } g(f(i)) = i \\ * & \text{otherwise} \end{cases}$$

Given premise pointer, can evaluate $h(i)$ with $O(\log n)$ queries to Boolean variables defining f and g in $\lceil PHP \rceil$

Note that if $h(i) = *$, then we have witnessed violated axiom in $\lceil PHP \rceil$

h can easily be inverted


¬PHP
HARD CODED
⊥
x_1
$\bar{x}_1$
$x_1 \vee x_2$
$x_1 \vee \bar{x}_2$
$\bar{x}_1 \vee x_2$
$\bar{x}_1 \vee \bar{x}_2$
¬PHP
$x_1 \bar{x}_2 x_3$
$x_1 x_2 \bar{x}_3$
$x_1 \bar{x}_2 \bar{x}_3$
¬PHP
x_1 $x_2 \bar{x}_3 x_4$
$x_1 \bar{x}_2 x_3 \bar{x}_4$
x_1 $x_2 \bar{x}_3 \bar{x}_4$
¬PHP
x_1 $\bar{x}_2 x_3 \bar{x}_4 x_5$
$x_1 x_2 \bar{x}_3 x_4 \bar{x}_5$
x_1 $\bar{x}_2 x_3 \bar{x}_4 \bar{x}_5$

Consider jth block B at full-size layer L_i

Premise pointers $h(2j-1)$ and $h(2j)$

Axiom pointers only used at layer L_n (if $i < n$ determined as below)

Literal indicators Determined in same way as literal pointers

Compute h^{-1} repeatedly to generate path to $L_{i-1}, L_{i-2}, \ldots,$ until hit block without pointer to resolvent.

Case 1: $B_5 = \perp$ WO reached

Mark B as disabled ∅

Case 2: $B_5 = \perp$ reached — Then linear depth i

By tracing left-right turns in path can figure out signs of literals over $x_1, x_2, \ldots, x_i$

For every j^*th block in layer i^*

$$h(2j^*-1) \longleftrightarrow x_{i^*}$$
$$h(2j^*) \longleftrightarrow \overline{x_{i^*}}$$

Depth of decision tree for literals $O(n \log n)$

Resolved variable pointer Once we know what layer L_i we are at, this is x_{i+1}

Block type disabled if sth goes wrong (or for left-over block)

desired if layer $i < n$

axiom if layer $i = n$

This reduction is block-aware!
A single decision tree in depth $O(n \log n)$ can compute all of these variables.

Also need to show:
For any clause $C \in Ref(F, s)$
all clauses in CNF representation of C of $\neg PHP$ are implied by some axiom of $\neg PHP$

Contrapositive: Consider violation of clause in CNF representation $\Rightarrow$ find violated clause in $\neg PHP$

Disabled blocks can never have violated clauses

Axiom blocks are pointing to the correct axioms for which they are weakenings

For derived blocks, literal sets and resolved variables are computed consistently

But for premises we can have
$h(2j-1) = *$ or $h(2j) = *$

But, as discussed above, this can only happen when axiom of $\neg PHP$ violated, since we found i^*, j^* s.t. $f(i^*) = j^*$ but $g(j^*) \neq i^*$

This concludes proof of Lemma 7 ∎

(For TreeRef(F,s), we can compute resolvent pointers by inverting premise pointers.)

Upper bound in part (i) in Lemma 3

True for regular resolution = read-once branching program (ROBP)

Fix assignment α satisfying F

Start at $B_s = \bot$

Walk backwards in proof to axiom block

Maintain invariant

> "Current block is enabled and encodes clause falsified by α"

Since this will fail at the latest for axiom block, ROBP identifies falsified clause in Ref(F, s) given assignment β to Vars(Ref(F,s))

Define ROBP Π as having special nodes

V_i
V_i^{axiom}
} for $i = s, s-1, \ldots, 1$

Suppose α sets $\{l_1, \ldots, l_n\}$ to true

V_i remembers:

- B_i is enabled
- encodes clause not containing l_i, $i \in [n]$

v_i^{axiom} remembers:
- all that v_i remembers
- B_i is an axiom block

Structure of Π

- Start by querying B_s
  should be enabled block encoding $\bot$ $\rightsquigarrow v_s$
  Otherwise done

- At v_i
  Query if B_i axiom block
  If so, go to v_i^{axiom}
  Otherwise query variable x resolved to get clause at B_i
  Find pointer to premise B_j containing literal over x falsified by α
  Check that premise B_j doesn't contain any other l_i, $i \in [n]$
  check $j < i$, of course
  Check B_j enabled
  Move to v_j

- At v_i^{axiom}
  Look up axiom $C_j \in F$
  C_j contains some l_i, $i \in [n]$, since α satisfies C_j
  Leads to violation

Non-automatability for Nullstellensatz and polynomial calculus:

Can define low-degree reductions

Reduce from rPHP to TreeRef(F,s)

Use [Impagliazzo-Pudlák-Sgall '99] to turn degree lower bound to size lower bound for PC

$\Rightarrow$ lower bound also for NS

Upper bound (for Nullstellensatz): Use that TreeRef(F,s) has more constraints.

Sweeping all details under the rug...

Non-automatability also shown for cutting planes [GKMP '20]

Open for
- Sherali-Adams
- Sum-of-squares (PHP not hard)

Not known for CNF formulas for bounded-depth Frege except under (much) stronger assumptions