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