Topics in propositional proof complexity

Leszek Kołodziejczyk

PhD Open
Warsaw, January 2021

What is proof complexity about?

(Propositional) proof complexity attempts to understand how difficult it is to prove formulas in various *proof systems* for propositional logic.

Usually, the main goal is to obtain lower bounds on one of a number of measures of the complexity of a proof.

What is proof complexity about?

(Propositional) proof complexity attempts to understand how difficult it is to prove formulas in various *proof systems* for propositional logic.

Usually, the main goal is to obtain lower bounds on one of a number of measures of the complexity of a proof.

Definition

- TAUT is the set of all tautological DNF's.
- A propositional proof system is a polytime computable surjection from $\{0,1\}^*$ onto TAUT.

These lectures

- A selection of ideas and results in proof complexity: some classical, some new.
- Choice strongly dependent on personal taste of the lecturer. Here, this means a preference for arguments inspired by an (often very simple) idea from logic.
- Avoids technically or conceptually advanced arguments (switching lemmas, expanders, communication complexity, bounded arithmetic...).

Motivation 1: NP vs coNP

Definition

A proof system P is p-bounded
if every tautology τ has a proof in P of size $|\tau|^{O(1)}$.

Theorem

A p-bounded proof system exists iff NP = coNP.

Proof.

($\Rightarrow$) TAUT is coNP-complete, and a p-bounded P would mean:
$\tau \in \text{TAUT} \leftrightarrow \exists\pi(|\pi| \leq |\tau|^{O(1)} \wedge \pi$ is a P-proof of $\tau)$.
($\Leftarrow$) Build a proof system in which proofs are accepting computations of the nondeterministic polytime machine recognizing TAUT. $\square$

Motivation 1: NP vs coNP

Definition

A proof system P is p-bounded
if every tautology τ has a proof in P of size $|\tau|^{O(1)}$.

Theorem

A p-bounded proof system exists iff NP = coNP.

This inspired Cook's programme for proving NP ≠ coNP:
let's prove superpolynomial lower bounds on proof size
for stronger and stronger proof systems, until eventually...

So far: interesting results, but not even close to the original goal.

Examples of proof systems 1: Frege systems

Typical reasonable "textbook" systems, such as:

(1) a *Hilbert-style* system,
with finitely many axioms, e.g. $(\neg F \to F) \to F$,
and rules (like modus ponens: "from $F \to G$ and F derive G").

Examples of proof systems 1: Frege systems

Typical reasonable "textbook" systems, such as:

(2) *sequent calculus*, with lines of the form $\Gamma \vdash \Delta$
where Γ, Δ finite lists of formulas interpreted as $\bigwedge, \bigvee$ respectively.
Many rules, including so-called cut rule – analogue of modus ponens.

Examples of proof systems 1: Frege systems

Typical reasonable "textbook" systems, such as:

(3) a particularly simple example: a *Tait-style* system.
Connectives only $\wedge, \vee$, plus negation only at variables, written as $\overline{x}$;
negation defined using De Morgan laws beyond that.
Lines are finite sets of formulas, interpreted as disjunctions.

$$\frac{}{\Gamma, x, \overline{x}}\ \text{(axiom)} \qquad \frac{\Gamma}{\Gamma, F}\ \text{(weakening)}$$

$$\frac{\Gamma, F}{\Gamma, F \vee G}\ (\vee_1) \qquad \frac{\Gamma, G}{\Gamma, F \vee G}\ (\vee_2)$$

$$\frac{\Gamma, F \qquad \Gamma, G}{\Gamma, F \wedge G}\ (\wedge) \qquad \frac{\Gamma, F \qquad \Gamma, \overline{F}}{\Gamma}\ \text{(cut)}$$

Examples of proof systems 1: Frege systems

Originally, only Hilbert-style proof systems were called Frege systems.
But any two of the above are pairwise equivalent w.r.t. p-simulation:
in each direction, there is a polytime translation of a proof of
a DNF formula F in one system into a proof of F in the other.

Examples of proof systems 2: Extended Frege

If we extend a Frege system by a rule allowing us to abbreviate formulas by deriving

$$x \leftrightarrow F,$$

for any formula F and a fresh variable x, we get an Extended Frege system. Proof size in Extended Frege $\approx$ number of steps in Frege.

These are not expected to be p-simulated by Frege.

Examples of proof systems 3: bounded-depth Frege

Depth of a fla = number of alternations of $\wedge$'s and $\vee$'s in syntactic tree. E.g. literals have depth 0, CNF's and DNF's have depth ≥ 2.

A depth-d Frege system restricts a Frege system to allow only the use of formulas with depth $\leq d$.

This is sensitive to small changes in definitions. But family of bounded-depth systems $= \bigcup_{d \in \mathbb{N}} \{\text{depth-}d \text{ system}\}$ is quite robust.

These are known not to p-simulate Frege.

Examples of proof systems 4: Resolution

A system for *refuting* CNF's instead of proving DNF's.
Roughly, depth-0 Frege viewed as a refutation system.

Refutation lines are clauses = finite sets of literals,
regarded as $\bigvee$'s. So, set of clauses $\approx$ CNF.

Resolution rule:

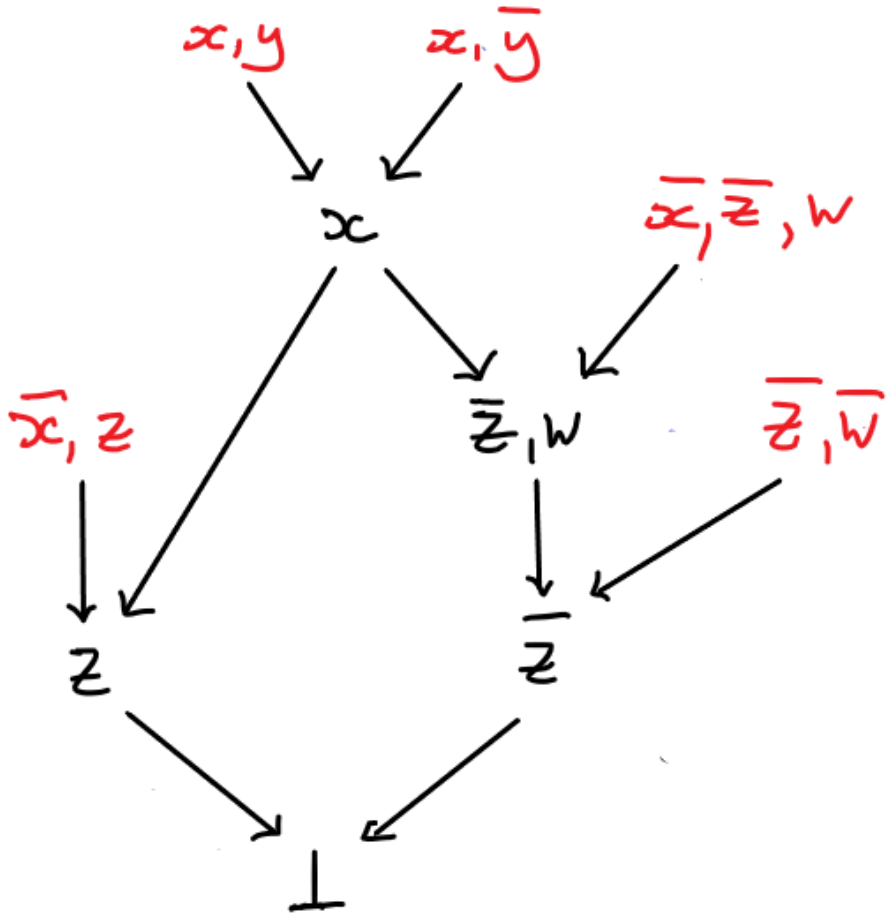
$$\frac{C, x \qquad D, \overline{x}}{C, D}$$

Sometimes weakening also allowed, nothing else.
Refutation = derivation of empty clause $\bot$.

A Resolution refutation

A refutation of $F := (x \vee y) \wedge (x \vee \bar{y}) \wedge (\bar{x} \vee z) \wedge (\bar{x} \vee \bar{z} \vee w) \wedge (\bar{z} \vee \bar{w})$:

1. $x \vee y$
2. $x \vee \bar{y}$
3. x [1,2]
4. $\bar{x} \vee z$
5. z [3,4]
6. $\bar{x} \vee \bar{z} \vee w$
7. $\bar{z} \vee w$ [3,6]
8. $\bar{z} \vee \bar{w}$
9. $\bar{z}$ [7,8]
10. $\perp$ [5,9]

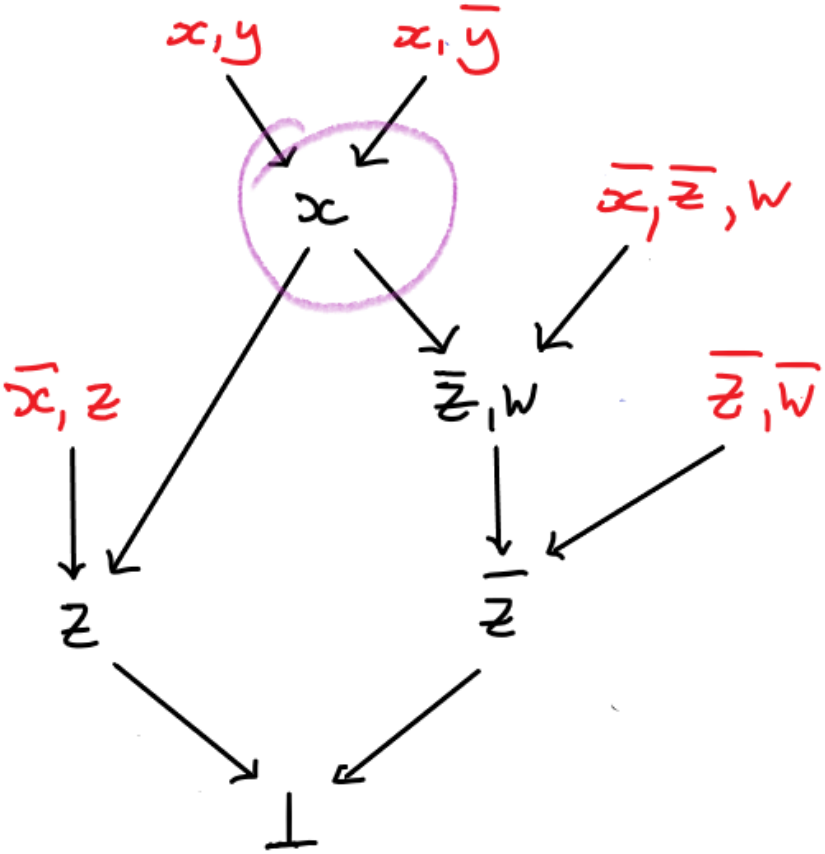


Clauses of F (= initial clauses) in red.

Treelike vs daglike Resolution

In treelike Resolution, any non-initial clause can only be used once as a premise. If needed again, it has to be rederived.

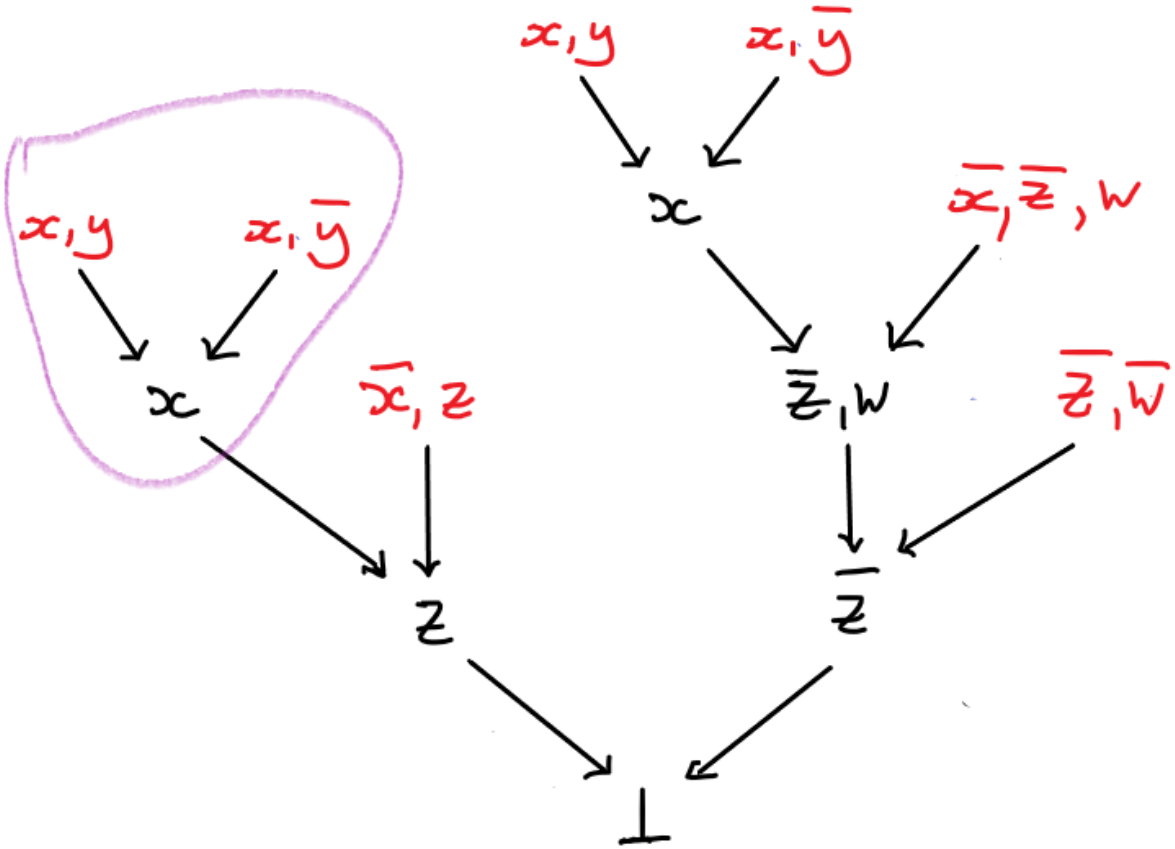
This corresponds to the proof dag being a tree.



Treelike vs daglike Resolution

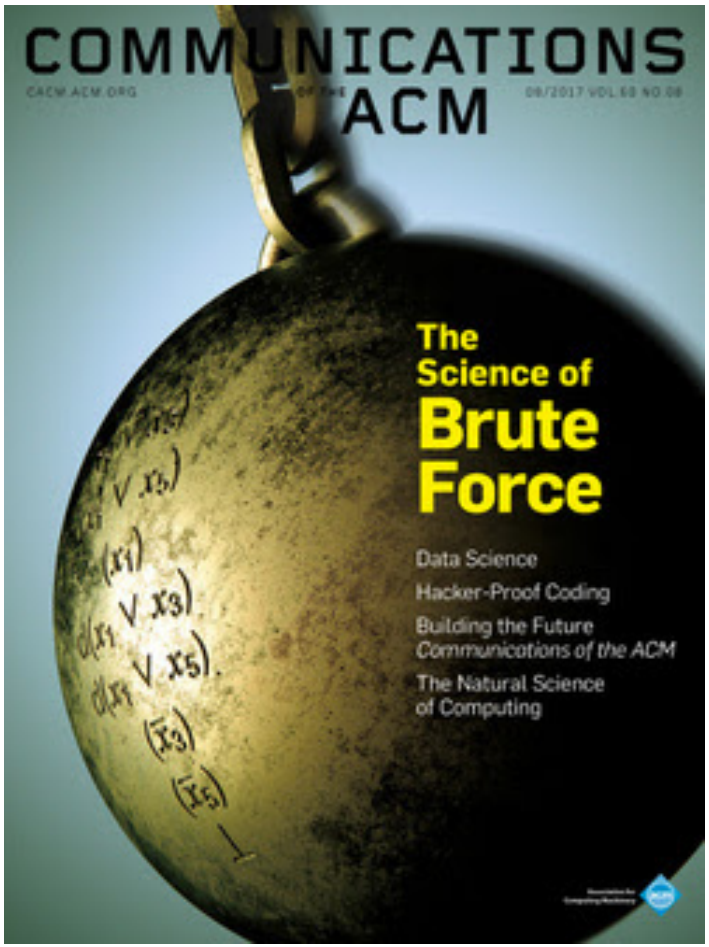
In treelike Resolution, any non-initial clause can only be used once as a premise. If needed again, it has to be rederived.

This corresponds to the proof dag being a tree.



Motivation 2: SAT-solvers

Relatively weak proof systems (especially Resolution)
are a good theoretical model of state-of-the-art SAT-solvers.



Motivation 2: SAT-solvers

The core of the classical DPLL algorithm from the 1960's:

procedure DPLL(F, ρ) ▷ F formula, ρ partial assignment
  if ρ satisfies each clause of F **output** ρ and **terminate**;
  if ρ falsifies some of F **return** False;
  choose a literal x; ▷ x is the decision literal
  call DPLL$(F, \rho[x:=0])$;
  call DPLL$(F, \rho[x:=1])$;
  return False;

Run of DPLL on unsat F produces treelike Res refutation of F.
Otoh, given treelike Res refutation and "guessing" right decision vars, DPLL can perform DFS of the refutation and return False.

$\rightsquigarrow$ Treelike Resolution $\approx$ "nondeterministic" DPLL.

Motivation 2: SAT-solvers

Modern CDCL (*conflict-driven clause learning*) solvers also include:

- unit propagation: if some clause has just one literal ℓ not falsified in ρ, it is automatically set to true,
- clause learning: each time the solver finds that ρ falsifies F, it learns a "conflict clause" C (easily derivable from F in Resolution) that caused this, and adds C to F,
- restarts: the solver can choose to forget all of the assignment ρ but keep the learned clauses.

All this can still be modelled by Resolution. Otoh, given arbitrary Res refutation and "guessing" right decision vars and restart moments, CDCL can learn all clauses in the refutation. [AFT13, PD13].

$\rightsquigarrow$ Resolution $\approx$ "nondeterministic" CDCL.

Examples of proof systems 5: Polynomial Calculus

Lines are $p(\vec{x})$ for p polynomials over fixed field $\mathbb{F}$, mean $p(\vec{x}) = 0$.
This lets us express clauses: $x \vee \bar{y} \vee \bar{z} \mapsto (1 - x)yz$.

$$\frac{}{x^2 - x}\ \text{(boolean axiom)} \qquad \frac{p \qquad q}{ap + bq}\ \text{(addition)} \qquad \frac{p}{xp}\ \text{(multiplication)}$$

Refutation = derivation of the constant polynomial 1.

To fit the definition of proof system, $\mathbb{F}$ should be effectively given, like finite fields, $\mathbb{Q}$ etc. But most size lower bounds work for any field.

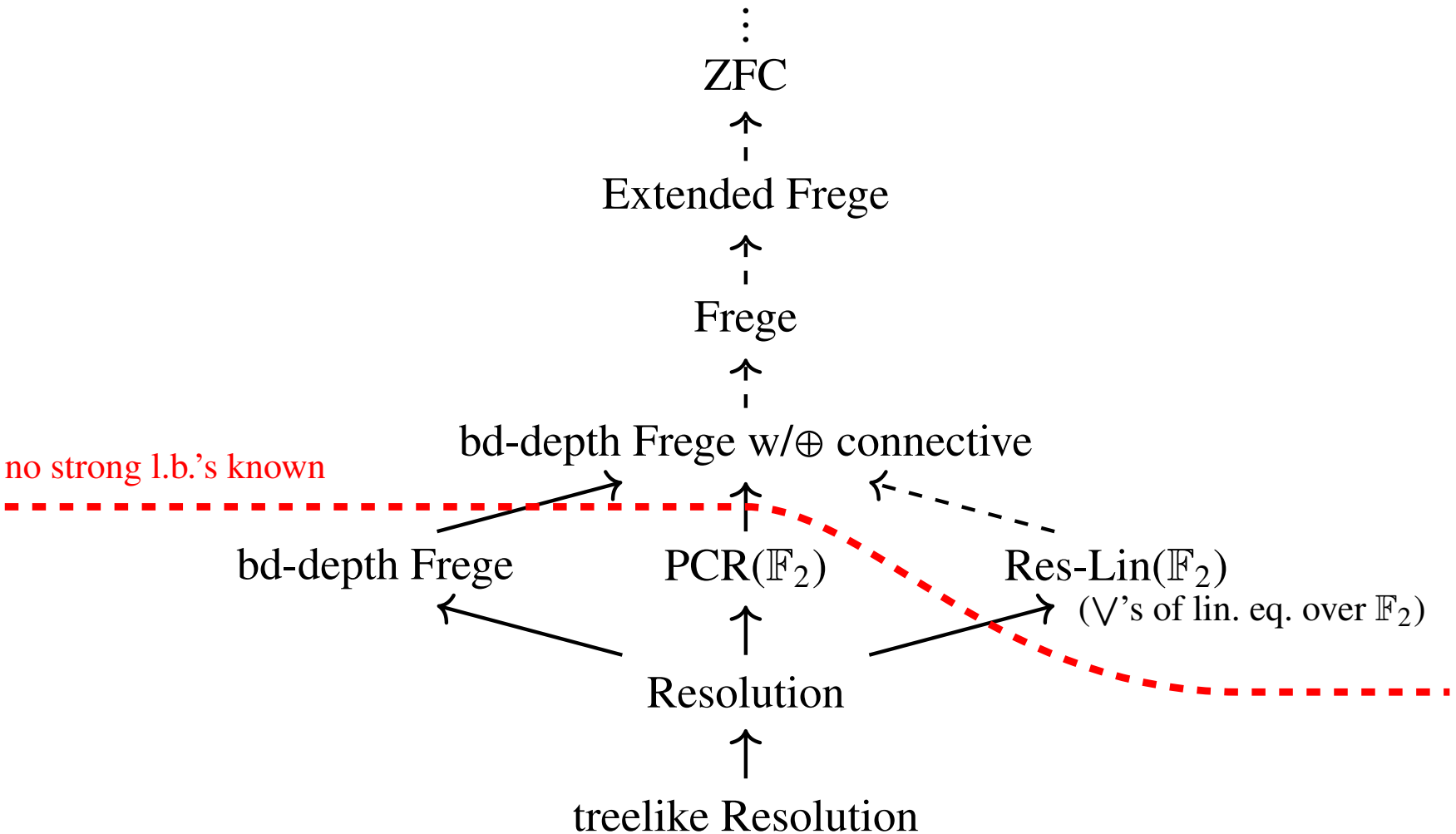
To p-simulate Resolution, one can add formal variables $\bar{x}$ and axioms $1 - x - \bar{x}$. This gives Polynomial Calculus with Resolution (PCR).

Other proof systems

- Any consistent theory T with polytime-recognizable axiomatization that can make sense of finite strings, e.g. ZFC. Proof of τ is a proof in T of "τ is a tautology".
- Systems that operate using linear inequalities with integer coefficients (Cutting Planes).
- Or using polynomial inequalities (Sum-of-Squares etc.).
- ...

Of the systems we defined, strong size lower bounds are known for Resolution, Polynomial Calculus (& PCR), and bounded-depth Frege. No superpolynomial bounds known for Frege (and some systems well below that).

Small part of proof complexity world, pictured



Plan for the lectures

- Tutorial: how to prove *a* Resolution size lower bound.
- Automatability of proof systems.
- Space of proofs and its connection to other measures.

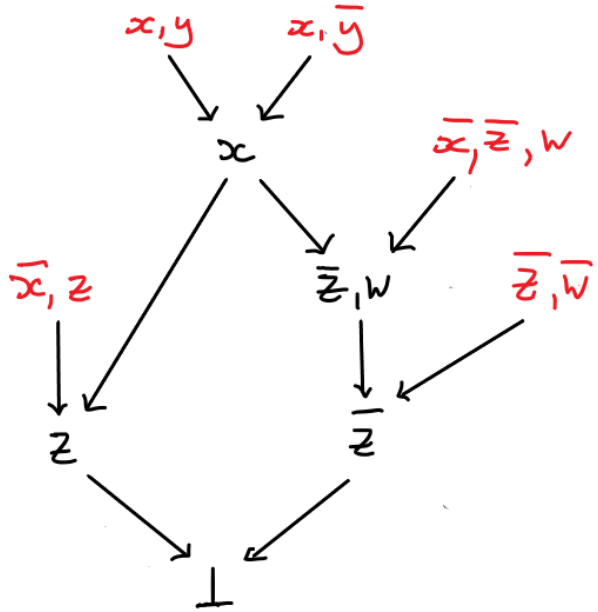
Plan for the lectures

- Tutorial: how to prove *a* Resolution size lower bound.
- Automatability of proof systems.
- Space of proofs and its connection to other measures.

Resolution width

The width of a clause is the number of literals in it.
The width of a Res refutation is the largest width of a clause in it.

E.g. our running example



has width 3, but only because of the initial clause $\overline{x} \vee \overline{z} \vee w$.

Resolution width

The width of a clause is the number of literals in it.
The width of a Res refutation is the largest width of a clause in it.

For a CNF F, the "width needed to refute F in Resolution", $W_R(F)$, is the smallest width of a Resolution refutation of F.

Proving lower bounds on $W_R(F)$ is sometimes easy.

The Atserias-Dalmau pebble game

The game is determined by CNF F and number of pebbles k.
There are two players: Prover and Duplicator.

Duplicator claims to know a satisfying assignment for F.
(In interesting cases, F is unsatisfiable.)

In each round:

- Prover picks up one of the k pebbles and places it on a vble of F.
- Duplicator responds with a 0/1 value for that variable.

Spoiler wins if at any point the assignment on the pebbled variables falsifies a clause of F. If that never happens, Duplicator wins.

Example: Atserias-Dalmau game with 2 pebbles

A play of the 2-pebble game on the formula

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

(a.k.a. $(x \not\leftrightarrow y) \wedge (x \leftrightarrow z) \wedge (y \leftrightarrow z)$) could start like this:

$$x \mapsto ?$$

Example: Atserias-Dalmau game with 2 pebbles

A play of the 2-pebble game on the formula

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

(a.k.a. $(x \not\leftrightarrow y) \wedge (x \leftrightarrow z) \wedge (y \leftrightarrow z)$) could start like this:

$$x \mapsto 1$$

Example: Atserias-Dalmau game with 2 pebbles

A play of the 2-pebble game on the formula

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

(a.k.a. $(x \not\leftrightarrow y) \wedge (x \leftrightarrow z) \wedge (y \leftrightarrow z)$) could start like this:

$$x \mapsto 1$$
$$y \mapsto ?$$

Example: Atserias-Dalmau game with 2 pebbles

A play of the 2-pebble game on the formula

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

(a.k.a. $(x \not\leftrightarrow y) \wedge (x \leftrightarrow z) \wedge (y \leftrightarrow z)$) could start like this:

$$x \mapsto 1$$
$$y \mapsto 0$$

Example: Atserias-Dalmau game with 2 pebbles

A play of the 2-pebble game on the formula

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

(a.k.a. $(x \not\leftrightarrow y) \wedge (x \leftrightarrow z) \wedge (y \leftrightarrow z)$) could start like this:

$$y \mapsto 0$$
$$z \mapsto ?$$

Example: Atserias-Dalmau game with 2 pebbles

A play of the 2-pebble game on the formula

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

(a.k.a. $(x \not\leftrightarrow y) \wedge (x \leftrightarrow z) \wedge (y \leftrightarrow z)$) could start like this:

$$y \mapsto 0$$
$$z \mapsto 0$$

Example: Atserias-Dalmau game with 2 pebbles

A play of the 2-pebble game on the formula

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

(a.k.a. $(x \not\leftrightarrow y) \wedge (x \leftrightarrow z) \wedge (y \leftrightarrow z)$) could start like this:

$$x \mapsto ?$$

$$z \mapsto 0$$

Example: Atserias-Dalmau game with 2 pebbles

A play of the 2-pebble game on the formula

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

(a.k.a. $(x \not\leftrightarrow y) \wedge (x \leftrightarrow z) \wedge (y \leftrightarrow z)$) could start like this:

$$x \mapsto 0$$

$$z \mapsto 0$$

Example: Atserias-Dalmau game with 2 pebbles

A play of the 2-pebble game on the formula

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

(a.k.a. $(x \not\leftrightarrow y) \wedge (x \leftrightarrow z) \wedge (y \leftrightarrow z)$) could start like this:

...

Atserias-Dalmau games and width

Theorem (Atserias-Dalmau 2008)

Let F be a k-CNF and let $w > k$. Then: $W_R(F) \geq w$ iff
Duplicator has a winning strategy in the w-pebble A-D game on F.

Atserias-Dalmau games and width

Theorem (Atserias-Dalmau 2008)

Let F be a k-CNF and let $w > k$. Then: $W_R(F) \geq w$ iff
Duplicator has a winning strategy in the w-pebble A-D game on F.

Proof.

($\Leftarrow$) A Res refutation π is a Prover strategy: go up π starting at $\bot$, always have pebbles on the variables in the clause C you are now at, and maintain the condition that Duplicator's assignment falsifies C.

For π of width $w-1$, this strategy uses up to w pebbles: extra pebble needed for the resolution variable in the inference used to derive C. So, if Duplicator wins w-pebble games, then $W_R(F) \geq w$.

We'll comment on ($\Rightarrow$) in class.

The pigeonhole principle

The most famous CNF in proof complexity!
PHP_n^m says that there is an injective multifunction from m into n.

Variables: x_{ij}, for $i \in [m], j \in [n]$. ▷ "Pigeon i goes to hole j".

Clauses:

$$x_{i1} \vee \ldots \vee x_{in}, \text{ for } i \in [m],$$
$$\overline{x_{ij}} \vee \overline{x_{i'j}}, \text{ for } i \neq i' \in [m], j \in [n].$$

This is "clearly" unsatisfiable when $m > n$.

We would like to prove a Res width lower bound of $\Omega(n)$,
but the problem is that PHP_n^m itself already has width n.

Narrow variants of PHP

We need a low-width reformulation of PHP_n^m.
There is more than one option:

Option 1: expanders

Take a bounded-degree bipartite expander graph on $[m] \sqcup [n]$ and keep only those variables x_{ij} where j is a neighbour of i in the graph.

Option 2: auxiliary variables

Add variables $y_{i1}, \ldots, y_{i(n-1)}$ for each i and replace $x_{i1} \vee \ldots \vee x_{in}$ with the 3-clauses $x_{i1} \vee y_{i1}$, $\overline{y_{i1}} \vee x_{i2} \vee y_{i2}$, ..., $\overline{y_{i(n-1)}} \vee x_{1n}$.

Our choice: bitwise PHP

Instead of having a separate variable for each pigeon and hole, specify the number of a hole by the bits in its binary notation.

Bitwise pigeonhole principle bitPHP_n^m

Variables: x_{ij}, for $i \in [m], j \in [\log n]$.
▷ Idea: values of $x_{i1}, \ldots, x_{i(\log n)}$ are the bits of the hole number to which pigeon i goes. Note that this makes the mapping of pigeons to holes single-valued by definition.

Clauses:

$$(x_{i1})^{b_1} \vee (x_{i'1})^{b_1} \vee \ldots \vee (x_{i(\log n)})^{b_{\log n}} \vee (x_{i'(\log n)})^{b_{\log n}},$$

for each $i \neq i' \in [m]$, $b_1, \ldots, b_{\log n} \in \{0, 1\}$.
▷ $(x)^1$ is x, $(x)^0$ is $\overline{x}$. So: two different pigeons don't both go to the hole whose number has bits $1 - b_1, \ldots, 1 - b_{\log n}$.

Bitwise pigeonhole principle bitPHP_n^m

Variables: x_{ij}, for $i \in [m], j \in [\log n]$.
▷ Idea: values of $x_{i1}, \ldots, x_{i(\log n)}$ are the bits of the hole number to which pigeon i goes. Note that this makes the mapping of pigeons to holes single-valued by definition.

Clauses:

$$(x_{i1})^{b_1} \vee (x_{i'1})^{b_1} \vee \ldots \vee (x_{i(\log n)})^{b_{\log n}} \vee (x_{i'(\log n)})^{b_{\log n}},$$

for each $i \neq i' \in [m]$, $b_1, \ldots, b_{\log n} \in \{0, 1\}$.
▷ $(x)^1$ is x, $(x)^0$ is $\overline{x}$. So: two different pigeons don't both go to the hole whose number has bits $1 - b_1, \ldots, 1 - b_{\log n}$.

Theorem

$W_R(\text{bitPHP}_n^m) \geq n$ *for each* m *and* n.

Resolution width lower bound on bitwise PHP

Theorem

$W_R(\text{bitPHP}_n^m) \geq n$ *for each* m *and* n.

Proof.

In the n-pebble game, Duplicator uses the following strategy:
if the pebbles are on $x_{i_1 j_1}, \ldots, x_{i_k j_k}$, where $k \leq n$, maintain
an injection from $\{i_1, \ldots, i_k\}$ into holes and answer accordingly. □

Resolution width lower bound on bitwise PHP

Theorem

$W_R(\text{bitPHP}_n^m) \geq n$ *for each* m *and* n.

Proof.

In the n-pebble game, Duplicator uses the following strategy:
if the pebbles are on $x_{i_1 j_1}, \ldots, x_{i_k j_k}$, where $k \leq n$, maintain
an injection from $\{i_1, \ldots, i_k\}$ into holes and answer accordingly. □

Remark

This proof also shows that a Resolution refutation of bitPHP_n^m
must contain a clause that mentions $\geq n$ pigeons.

Resolution width lower bound on bitwise PHP

Theorem

$W_R(\text{bitPHP}_n^m) \geq n$ *for each* m *and* n.

Proof.

In the n-pebble game, Duplicator uses the following strategy:
if the pebbles are on $x_{i_1 j_1}, \ldots, x_{i_k j_k}$, where $k \leq n$, maintain
an injection from $\{i_1, \ldots, i_k\}$ into holes and answer accordingly. □

Remark

This proof also shows that a Resolution refutation of bitPHP_n^m
must contain a clause that mentions $\geq n$ pigeons.

But does this have anything to do with bounds on refutation size?

The width-size connection for Resolution

The length of a refutation is the number of clauses in it.
Note: Length $\leq$ Size $\leq O($Length $\cdot$ Number of variables$)$.

$L_R(F)$ is the minimal length of a Resolution refutation on F.

Theorem (Ben-Sasson–Wigderson 2001)

For any unsatisfiable k-CNF formula F,

$$L_R(F) \geq \exp\left(\Omega\left(\frac{(W_R(F)-k)^2}{\textit{no. of vars in } F}\right)\right).$$

The width-size connection for Resolution

The length of a refutation is the number of clauses in it.
Note: Length $\leq$ Size $\leq O($Length $\cdot$ Number of variables$)$.

$L_R(F)$ is the minimal length of a Resolution refutation on F.

Theorem (Ben-Sasson–Wigderson 2001)

For any unsatisfiable k-CNF formula F,

$$L_R(F) \geq \exp\left(\Omega\left(\frac{(W_R(F)-k)^2}{\text{no. of vars in } F}\right)\right).$$

bitPHP_n^m is a $(\log n)$-CNF with $m \cdot \log n$ variables, so this would give e.g. an $2^{n/\log n}$ length lower bound on refutations of bitPHP_n^{n+1}.

We will not prove the BW2001 result.
Instead, we will prove a length lower bound for another formula.

Relativizing pigeonhole principles

In logic, given a structure $\mathbb{A}$ with a predicate U, we can talk about the structure $U^{\mathbb{A}}$ by relativizing quantifiers: $\forall x\,(U(x) \to \ldots)$ etc.

Relativized bitwise PHP, rel-bitPHP$_n^{m_1,m_2}$:

Variables: x_{ij}, for $i \in [m_2], j \in [\log n]$, and u_{ki}, for $k \in [m_1], i \in [m_2]$.
▷ Idea: u_{ki} means "pigeon i is the k-th element of U".

Clauses:

- $u_{k1} \vee \ldots \vee u_{km_2}$, for $k \in [m_1]$,
- $\overline{u_{ki}} \vee \overline{u_{k'i}}$, for $k \neq k' \in [m_1], i \in [m_2]$,
- $\overline{u_{ki}} \vee \overline{u_{k'i'}} \vee (x_{i1})^{b_1} \vee (x_{i'1})^{b_1} \vee \ldots \vee (x_{i(\log n)})^{b_{\log n}} \vee (x_{i'(\log n)})^{b_{\log n}}$,
  for $k \neq k' \in [m_1], i \neq i' \in [m_2], b_1, \ldots, b_{\log n} \in \{0,1\}$.

Size lower bound on bitwise PHP

Theorem

$L_R(\text{rel-bitPHP}_n^{n+1,2n}) \geq 2^{\Omega(n)}$.

Proof.

Size lower bound on bitwise PHP

Theorem

$L_R(\text{rel-bitPHP}_n^{n+1,2n}) \geq 2^{\Omega(n)}$.

Proof.

We will use random restrictions (partial assignments).
Given clause C and restriction ρ, $C\restriction_\rho$ is $\top$ if ρ satisfies a literal in ρ, and otherwise $C\restriction_\rho$ is C without the literals falsified by ρ.

What is the effect of such ρ on a refutation?

$$\frac{x, y \qquad \overline{x}, z}{y, z}$$

Size lower bound on bitwise PHP

Theorem

$L_R(\text{rel-bitPHP}_n^{n+1,2n}) \geq 2^{\Omega(n)}$.

Proof.

We will use random restrictions (partial assignments).
Given clause C and restriction ρ, $C\restriction_\rho$ is $\top$ if ρ satisfies a literal in ρ, and otherwise $C\restriction_\rho$ is C without the literals falsified by ρ.

What is the effect of such ρ on a refutation?

$$\frac{x, y \qquad \overline{x}, z}{y, z} \qquad x \mapsto 1$$

Size lower bound on bitwise PHP

Theorem

$L_R(\text{rel-bitPHP}_n^{n+1,2n}) \geq 2^{\Omega(n)}$.

Proof.

We will use random restrictions (partial assignments).
Given clause C and restriction ρ, $C\restriction_\rho$ is $\top$ if ρ satisfies a literal in ρ, and otherwise $C\restriction_\rho$ is C without the literals falsified by ρ.

What is the effect of such ρ on a refutation?

$$\frac{\cancel{\bar{x}},z}{y,z} \qquad\qquad x \mapsto 1$$

It remains a refutation, possibly with some weakening inferences!

Size lower bound on bitwise PHP

Theorem

$L_R(\text{rel-bitPHP}_n^{n+1,2n}) \geq 2^{\Omega(n)}$.

Proof.

Our restrictions ρ:

- Choose uniformly at random an injection h from $[n+1]$ to $[2n]$. Set $\rho(u_{ki}) = 1$ if $h(k) = i$ and $\rho(u_{ki}) = 0$ otherwise.
- For $i \notin \mathrm{Rg}(h)$, independently for each j set $\rho(x_{ij})$ by flipping a coin.
- Leave all variables x_{ij} for $i \in \mathrm{Rg}(h)$ unassigned.
- Note: $\text{rel-bitPHP}_n^{2n,n+1}\restriction_\rho \simeq \text{bitPHP}_n^{n+1}$!
  What can happen to a modest-size refutation π under ρ?

Size lower bound on bitwise PHP

Theorem

$L_R(\text{rel-bitPHP}_n^{n+1,2n}) \geq 2^{\Omega(n)}$.

Proof.

Remarks on pigeonhole principles

- Essentially all forms of PHP are hard for Resolution...
- ...and easy for Frege.
- Most forms remain hard for PCR (over any field). However, a variant that says "there is a bijection between $[n+1]$ and $[n]$" is easy.
- PHP_n^{n+1} (even the bijective variant) remains exponentially hard for bounded-depth Frege.
- But PHP_n^{2n} has a quasipolynomial-size proof in bounded-depth (in fact, in a "depth-(0.5)" refutation system).

Open problem

Does PHP_n^{2n} or $\mathrm{PHP}_n^{n^2}$ have polysize bounded-depth proofs?

Plan for the lectures

- Tutorial: how to prove *a* Resolution size lower bound.
- Automatability of proof systems.
- Space of proofs and its connection to other measures.

Automatability

Definition

A proof system P is automatable if there exists a procedure which, given $\tau \in \mathrm{TAUT}$, outputs a P-proof of τ in time $\mathrm{poly}(|\tau|, s)$, where s is the size of the smallest proof of τ.
P is weakly automatable if some automatable Q p-simulates P.

Automatability

Definition

A proof system P is automatable if there exists a procedure which, given $\tau \in \text{TAUT}$, outputs a P-proof of τ in time $\text{poly}(|\tau|, s)$, where s is the size of the smallest proof of τ.
P is weakly automatable if some automatable Q p-simulates P.

This would be a desirable property,
but it seems most systems are unlikely to have it.
How to show that proof systems are (not) automatable?

Feasible interpolation

If $F(\vec{x},\vec{y}) \wedge G(\vec{x},\vec{z})$ is unsatisfiable, then there exists some $I(\vec{x})$ (interpolant) such that $\vDash I(\vec{x}) \to \neg F(\vec{x},\vec{y})$ and $\vDash \neg I(\vec{x}) \to \neg G(\vec{x},\vec{z})$.

A priori, all such formulas $I(\vec{x})$ could be very large.

Feasible interpolation

If $F(\vec{x},\vec{y}) \wedge G(\vec{x},\vec{z})$ is unsatisfiable, then there exists some $I(\vec{x})$ (interpolant) such that $\vDash I(\vec{x}) \to \neg F(\vec{x},\vec{y})$ and $\vDash \neg I(\vec{x}) \to \neg G(\vec{x},\vec{z})$.
A priori, all such formulas $I(\vec{x})$ could be very large.

Definition

A proof system P has feasible interpolation if, for any P-proof π of $\neg(F(\vec{x},\vec{y}) \wedge G(\vec{x},\vec{z}))$, there is a $\mathrm{poly}(|\pi|)$-size circuit $I(\vec{x})$ interpolating between F and G.

Feasible interpolation

If $F(\vec{x},\vec{y}) \wedge G(\vec{x},\vec{z})$ is unsatisfiable, then there exists some $I(\vec{x})$ (interpolant) such that $\vDash I(\vec{x}) \to \neg F(\vec{x},\vec{y})$ and $\vDash \neg I(\vec{x}) \to \neg G(\vec{x},\vec{z})$.
A priori, all such formulas $I(\vec{x})$ could be very large.

Definition

A proof system P has feasible interpolation if, for any P-proof π of $\neg(F(\vec{x},\vec{y}) \wedge G(\vec{x},\vec{z}))$, there is a $\mathrm{poly}(|\pi|)$-size circuit $I(\vec{x})$ interpolating between F and G.

Observation

P has f.i. $\Rightarrow$ P has no small proofs of $\neg(F \wedge G)$ if no small I exists.

[But existence of such F, G requires some assumption from complexity:

say $\exists$OWP; at minimum, NP $\not\subseteq$ P/poly.]

Theorem

"Reasonable" P has no f.i. $\Rightarrow$ P not weakly automatable.

Proving non-automatability using lack of f.i.

(Bonet et al. 2000) proved that Frege systems do not have f.i.
assuming polysize circuits cannot factor Blum integers
(products of two primes of the form $4k + 3$);
so, Frege is not weakly automatable under the same assumption.

This can be pushed down to bounded-depth Frege (Bonet et al. 2004)
if the factoring above remains hard even for subexponential circuits.

Proving non-automatability using lack of f.i.

(Bonet et al. 2000) proved that Frege systems do not have f.i.
assuming polysize circuits cannot factor Blum integers
(products of two primes of the form $4k + 3$);
so, Frege is not weakly automatable under the same assumption.

This can be pushed down to bounded-depth Frege (Bonet et al. 2004)
if the factoring above remains hard even for subexponential circuits.

But:

Theorem (Krajíček 1997)

Resolution has feasible interpolation.

Proof.

Maybe class?!

Monotone feasible interpolation

Side point:

Theorem (Krajíček 1997)

Resolution also has monotone feasible interpolation:
if $F(\vec{x},\vec{y}) \wedge G(\vec{x},\vec{z})$ has a small Res refutation,
and $\vec{x}$ appears only negatively in F or only positively in G,
then there is a small monotone (w/o $\neg$'s) interpolating circuit $I(\vec{x})$.

(Razborov 1985) proved strong lower bounds on monotone circuits. This can be used to give lower bounds on Res refutations of "n-vertex graph coded by $\vec{x}$ is k-colourable and contains $(k+1)$-clique".

Automatability and Resolution

No hope for an argument via lack of f.i.

Automatability and Resolution

No hope for an argument via lack of f.i.

Theorem (Alekhnovich-Razborov '08 + Eickmeyer et al. '08)

Resolution is not automatable unless FPT = W[P].

But the proof also works for treelike Res, which is quasipolynomial-time automatable! (cf. Beame-Pitassi 1996).

Automatability and Resolution

No hope for an argument via lack of f.i.

Theorem (Alekhnovich-Razborov '08 + Eickmeyer et al. '08)

Resolution is not automatable unless FPT = W[P].

But the proof also works for treelike Res, which is quasipolynomial-time automatable! (cf. Beame-Pitassi 1996).

Theorem (Atserias-Müller 2019)

Resolution is not automatable unless P=NP.

This time, the result is specific to daglike Res and generalizes to quasipolynomial etc. time under appropriate change in assumption.

Main goal for today: (more or less) proving this theorem.

Refutation formulas

Basic idea:

Given CNF formula F and a number $s \in \mathbb{N}$, define another CNF $\mathrm{REF}(F, s)$ that says F has Res refutation of length s.

Then [Pudlák 2003]:

$F \in \mathrm{SAT} \Rightarrow \mathrm{REF}(F, s)$ has small Res refutation.

Hope:

$F \notin \mathrm{SAT} \Rightarrow \mathrm{REF}(F, s)$ has no small Res refutation.

Refutation formulas

Basic idea:

Given CNF formula F and a number $s \in \mathbb{N}$, define another CNF $\mathrm{REF}(F, s)$ that says F has Res refutation of length s.

Then [Pudlák 2003]:

$F \in \mathrm{SAT} \Rightarrow \mathrm{REF}(F, s)$ has small Res refutation.

Hope:

$F \notin \mathrm{SAT} \Rightarrow \mathrm{REF}(F, s)$ has no small Res refutation.

It is easier to get a width lower bound,
so we will work with a relativized version of $\mathrm{REF}(F, s)$...

Comments on Atserias-Müller

- This actually proves an inapproximability result:
  if P ≠ NP, then hard to approximate how large
  the smallest Res refutation is with subexponential error.
- [Garlík 2019]: Resolution size lower bounds on refutations
  of the original formulas REF(F, s).
- [de Rezende et al. 2020, Göös et al. 2020]: Generalization of
  non-automatizability to many other systems, including PCR.

Open problem

Is Resolution weakly automatizable?

- Would imply that parity games are in P [Beckmann et al. 2014].
- Equivalent to feasible interpolation for a system Res(2): like Res,
  but using 2-DNF's instead of clauses.
  [Res(2) has small proofs of the *reflection principle* for Res:
  "if $\vec{y}$ is a small Res refutation of CNF $\vec{x}$, then $\vec{z}$ is not a satisfying assignment for x".]

Plan for the lectures

- Tutorial: how to prove *a* Resolution size lower bound.
- Automatability of proof systems.
- Space of proofs and its connection to other measures.

Space in Resolution

Space of a proof ≈ how large a blackboard (or memory) is needed so that for each step of the proof, all the formulas required to verify it fit on the blackboard at the same time.

Space in Resolution

We can present each Res refutation π as a sequence of configurations $M_1, \ldots, M_t$, where each M_i is a list of clauses.

$$M_i = \boxed{C_1}\boxed{C_2}\boxed{C_3} \quad \cdots \quad \boxed{}\boxed{C_{s_i}}$$

- $M_1 = \varnothing$,
- $M_t = \{\bot\}$,
- M_{i+1} obtained from M_i by one of:
  - Axiom download: add a clause of the formula F being refuted,
  - Inference: add the effect of resolving two clauses from M_i,
  - Deletion: delete a clause appearing in M_i.

Space of π = largest s_i for M_i appearing in π.
$S_R(F)$ = smallest space of a refutation π of F.

Space vs other measures

Fact (Esteban-Torán 2001)

If F unsatisfiable with n variables, then $S_R(F) \leq n + O(1)$.

Theorem (Ben-Sasson–Nordström 2008)

There are CNF's $\{F_n\}_{n\in\mathbb{N}}$ s.t. $W_R(F_n) = O(1), S_R(F_n) = \Omega(n/\log n)$.

So, space can be much larger than width.
What about the other direction?

Space vs other measures

Fact (Esteban-Torán 2001)

If F unsatisfiable with n variables, then $S_R(F) \leq n + O(1)$.

Theorem (Ben-Sasson–Nordström 2008)

There are CNF's $\{F_n\}_{n\in\mathbb{N}}$ s.t. $W_R(F_n) = O(1), S_R(F_n) = \Omega(n/\log n)$.

So, space can be much larger than width.
What about the other direction?

Theorem (Atserias-Dalmau 2008)

For F a k-CNF, if F has a refutation in space s,
then it has one in width $s + k$. [So: $S_R(F) \geq W_R(F) - k$].

Res Space ≥ width - k

Proof.

- Assume that F has a Res refutation in space s: $M_1, \ldots, M_t$.
- Assume also that there is a winning strategy $\mathcal{H}$ for Duplicator in the $(s+k)$-pebble game.

Res Space ≥ width - k

Proof.

- Assume that F has a Res refutation in space s: $M_1, \ldots, M_t$.
- Assume also that there is a winning strategy $\mathcal{H}$ for Duplicator in the $(s+k)$-pebble game.

What is $\mathcal{H}$, really?
A nonempty family of partial assignments such that for each $\rho \in \mathcal{H}$:

(i) $|\rho| \leq s + k$,

(ii) if $\sigma \subseteq \rho$, then $\sigma \in \mathcal{H}$,

(iii) if $|\rho| < s + k$ and x vble, then there is $\mathcal{H} \ni \sigma \supseteq \rho$ with $x \in \mathrm{Dom}(\sigma)$,

(iv) ρ does not falsify any clause of F.

Res Space ≥ width - k

Proof.

- Assume that F has a Res refutation in space s: $M_1, \ldots, M_t$.
- Assume also that there is a winning strategy $\mathcal{H}$ for Duplicator in the $(s+k)$-pebble game.
- We can prove by induction on $i = 1, \ldots, t$ that there is $\rho_i \in \mathcal{H}$ with $|\rho_i| \leq s$ satisfying each clause in M_i.
- Induction step for Axiom download: use $|\rho_i| \leq s$ plus fact that no $\rho \in \mathcal{H}$ falsifies F to get assignment satisfying M_{i+1}. Then trim down to s bits because you only need s bits to satisfy s clauses.
- Other cases of induction step: trivial.
- But $\mathbb{M}_t$ contains $\bot$: contradiction. □

Polynomial Calculus Space

What about space for PCR?

Here each configuration is a list of polynomials, but for purposes of counting space we put each monomial in a separate cell.

$$M_i = \boxed{m_1}\boxed{m_2}\boxed{m_3} \quad \cdots \quad \boxed{\phantom{m_{s_i}}}\boxed{m_{s_i}}$$

Axiom download, Inference, Deletion as before,
though of course the class of inferences now wider:

$$\frac{}{x^2 - x} \qquad \frac{p \qquad q}{ap + bq} \qquad \frac{p}{xp} \qquad \frac{}{1 - x - \bar{x}}$$

Space of π = largest s_i for M_i appearing in π.
$S_{PCR}(F)$ = smallest space of a refutation π of F.

Bounding Polynomial Calculus Space

Space in PCR not too well understood. Only known to give linear improvement over space in Resolution. (And even this with caveats...)

The role of Res width is played in PCR by degree:

- Clauses with k vars get translated to degree k polys.
- There is a degree-size connection à la Ben-Sasson–Wigderson.

Question:

Can PCR space be lower-bounded by degree?

Context: $S_{PCR}(F[\oplus]) \geq \Omega(W_R(F) - k)$ [Filmus et al. 2013]

Bounding Polynomial Calculus Space

Space in PCR not too well understood. Only known to give linear improvement over space in Resolution. (And even this with caveats...)

The role of Res width is played in PCR by degree:

- Clauses with k vars get translated to degree k polys.
- There is a degree-size connection à la Ben-Sasson–Wigderson.

Question:

Can PCR space be lower-bounded by degree?

Context: $S_{PCR}(F[\oplus]) \geq \Omega(W_R(F) - k)$ [Filmus et al. 2013]

Theorem (Galesi et al. 2019)

For F a k-CNF, if F has a PCR refutation in space s, then it has a Res refutation in width $4s^2 + k$. [So: $S_{PCR}(F) \geq \Omega(\sqrt{W_R(F) - k})$].

Comments on space

- The proof of the PCR space/Res width link has little to do with PCR as such. Works for any proof system with sound rules such that each line is a boolean combination of $\leq s$ terms.
- Optimality of the construction not at all clear. Otoh, there are formulas for which known l.b. on S_{PCR} is $\sqrt{\ }$ of l.b. on S_R.

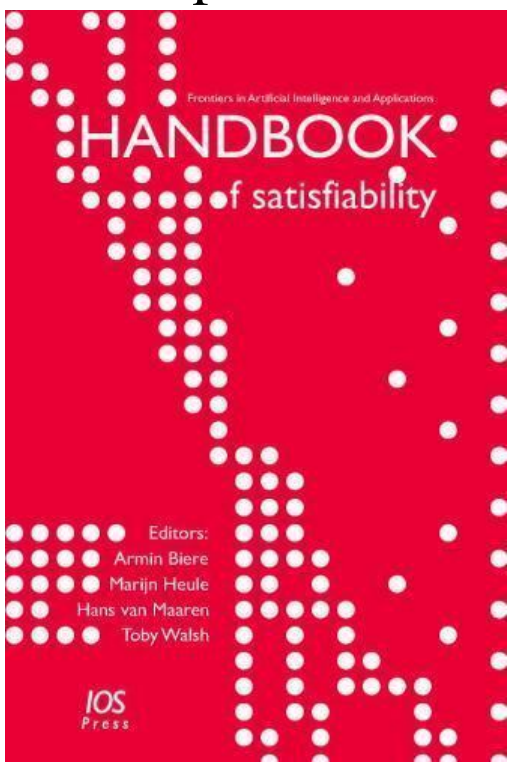
Open problem

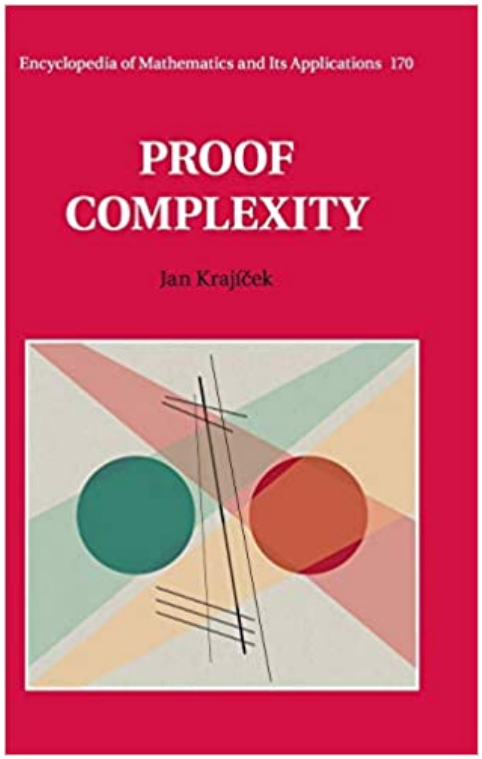
Is is the case that $S_{PCR}(F) \geq \Omega(W_R(F) - k)$?

- Not covered: tradeoffs between the measures. Often, it is possible to optimise a proof w.r.t. one measure or w.r.t. another, but not both simultaneously.

Thank you!

Chapter 7 of:





Upcoming online events: `simons.berkeley.edu/workshops`.