Short Proofs are Hard to Find

Ian Mertz

University of Toronto

Joint work w/ Toni Pitassi, Hao Wei

IAS, December 5, 2017

Proof complexity

$*54{\cdot}43$. $\vdash :. \alpha, \beta \in 1 \,.\, \supset : \alpha \cap \beta = \Lambda \,.\, \equiv \,.\, \alpha \cup \beta \in 2$

Dem.

$$\vdash \,.\, *54{\cdot}26 \,.\, \supset \vdash :. \alpha = \iota`x \,.\, \beta = \iota`y \,.\, \supset : \alpha \cup \beta \in 2 \,.\, \equiv \,.\, x \neq y \,.$$

$$[*51{\cdot}231] \qquad \equiv \,.\, \iota`x \cap \iota`y = \Lambda \,.$$

$$[*13{\cdot}12] \qquad \equiv \,.\, \alpha \cap \beta = \Lambda \qquad (1)$$

$$\vdash \,.\, (1) \,.\, *11{\cdot}11{\cdot}35 \,.\, \supset$$

$$\vdash :. (\exists x, y) \,.\, \alpha = \iota`x \,.\, \beta = \iota`y \,.\, \supset : \alpha \cup \beta \in 2 \,.\, \equiv \,.\, \alpha \cap \beta = \Lambda \qquad (2)$$

$$\vdash \,.\, (2) \,.\, *11{\cdot}54 \,.\, *52{\cdot}1 \,.\, \supset \vdash \,.\, \text{Prop}$$

From this proposition it will follow, when arithmetical addition has been defined, that $1 + 1 = 2$.

Proof complexity

$*54{\cdot}43.\quad \vdash :. \alpha, \beta \in 1 \,.\, \supset : \alpha \cap \beta = \Lambda \,.\, \equiv .\, \alpha \cup \beta \in 2$

Dem.

$\vdash . *54{\cdot}26 \,.\, \supset \vdash :. \alpha = \iota`x \,.\, \beta = \iota`y \,.\, \supset : \alpha \cup \beta \in 2 \,.\, \equiv .\, x \neq y \,.$

$[*51{\cdot}231] \qquad \equiv .\, \iota`x \cap \iota`y = \Lambda \,.$

$[*13{\cdot}12] \qquad \equiv .\, \alpha \cap \beta = \Lambda \qquad (1)$

$\vdash . (1) . *11{\cdot}11{\cdot}35 \,.\, \supset$

$\vdash :. (\exists x, y) \,.\, \alpha = \iota`x \,.\, \beta = \iota`y \,.\, \supset : \alpha \cup \beta \in 2 \,.\, \equiv .\, \alpha \cap \beta = \Lambda \qquad (2)$

$\vdash . (2) . *11{\cdot}54 \,.\, *52{\cdot}1 \,.\, \supset \vdash .$ Prop

From this proposition it will follow, when arithmetical addition has been defined, that $1 + 1 = 2$.

How long is the shortest $\mathcal{P}$-proof of τ?

Proof complexity

$*54{\cdot}43.\ \vdash :.\ \alpha, \beta \in 1 .\supset : \alpha \cap \beta = \Lambda . \equiv . \alpha \cup \beta \in 2$

Dem.

$\vdash . *54{\cdot}26 . \supset \vdash :. \alpha = \iota`x . \beta = \iota`y . \supset : \alpha \cup \beta \in 2 . \equiv . x \neq y .$

$[*51{\cdot}231] \qquad \equiv . \iota`x \cap \iota`y = \Lambda .$

$[*13{\cdot}12] \qquad \equiv . \alpha \cap \beta = \Lambda \qquad (1)$

$\vdash . (1) . *11{\cdot}11{\cdot}35 . \supset$

$\vdash :. (\exists x, y) . \alpha = \iota`x . \beta = \iota`y . \supset : \alpha \cup \beta \in 2 . \equiv . \alpha \cap \beta = \Lambda \qquad (2)$

$\vdash . (2) . *11{\cdot}54 . *52{\cdot}1 . \supset \vdash .$ Prop

From this proposition it will follow, when arithmetical addition has been defined, that $1 + 1 = 2$.

How long is the shortest $\mathcal{P}$-proof of τ?

Can we find short $\mathcal{P}$-proofs of τ?

Proof complexity

$*54{\cdot}43.\ \vdash :. \alpha, \beta \in 1 \,.\, \supset : \alpha \cap \beta = \Lambda \,.\, \equiv \,.\, \alpha \cup \beta \in 2$

Dem.

$\vdash . *54{\cdot}26 \,.\, \supset \vdash :. \alpha = \iota\text{‘}x \,.\, \beta = \iota\text{‘}y \,.\, \supset : \alpha \cup \beta \in 2 \,.\, \equiv \,.\, x \neq y \,.$

$[*51{\cdot}231] \qquad \equiv \,.\, \iota\text{‘}x \cap \iota\text{‘}y = \Lambda \,.$

$[*13{\cdot}12] \qquad \equiv \,.\, \alpha \cap \beta = \Lambda \qquad (1)$

$\vdash . (1) \,.\, *11{\cdot}11{\cdot}35 \,.\, \supset$

$\vdash :. (\exists x, y) \,.\, \alpha = \iota\text{‘}x \,.\, \beta = \iota\text{‘}y \,.\, \supset : \alpha \cup \beta \in 2 \,.\, \equiv \,.\, \alpha \cap \beta = \Lambda \qquad (2)$

$\vdash . (2) \,.\, *11{\cdot}54 \,.\, *52{\cdot}1 \,.\, \supset \vdash .$ Prop

From this proposition it will follow, when arithmetical addition has been defined, that $1 + 1 = 2$.

How long is the shortest $\mathcal{P}$-proof of τ?

Can we find short $\mathcal{P}$-proofs of τ?

Proof systems

Propositional proof system [Cook-Reckhow]

A *propositional proof system* is an onto map from proofs to tautologies checkable in polynomial time.

Proof systems

Propositional proof system [Cook-Reckhow]

A *propositional proof system* is an onto map from refutations to unsatisfiable formulas checkable in polynomial time.

Proof systems

Propositional proof system [Cook-Reckhow]

A *propositional proof system* is an onto map from refutations to unsatisfiable formulas checkable in polynomial time.

Polynomially-bounded PPS [Cook-Reckhow]

A PPS $\mathcal{P}$ is *polynomially bounded* if for every unsatisfiable k-CNF τ with n variables and $\text{poly}(n)$ clauses ($k = O(\log n)$), there exists a $\mathcal{P}$-proof π such that $|\pi| \leq \text{poly}(n)$.

Proof systems

Propositional proof system [Cook-Reckhow]

A *propositional proof system* is an onto map from refutations to unsatisfiable formulas checkable in polynomial time.

Polynomially-bounded PPS [Cook-Reckhow]

A PPS $\mathcal{P}$ is *polynomially bounded* if for every unsatisfiable k-CNF τ with n variables and poly(n) clauses ($k = O(\log n)$), there exists a $\mathcal{P}$-proof π such that $|\pi| \leq \text{poly}(n)$.

Theorem (Cook-Reckhow)

NP $=$ coNP *iff there exists a polynomially-bounded PPS.*

Resolution

Axioms:

$a \vee b \vee c$

$a \vee \bar{c}$

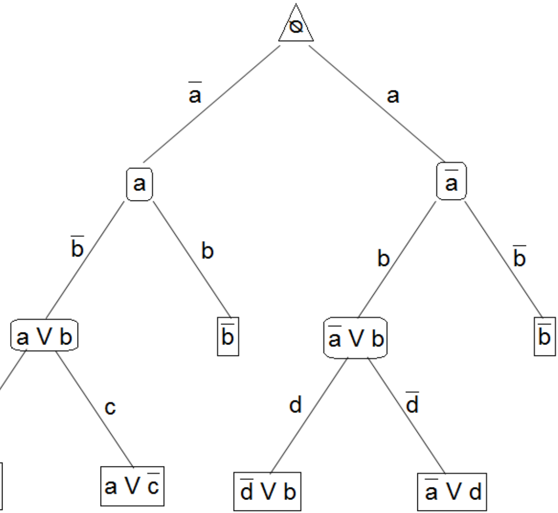
$\bar{a} \vee d$

$\bar{b}$

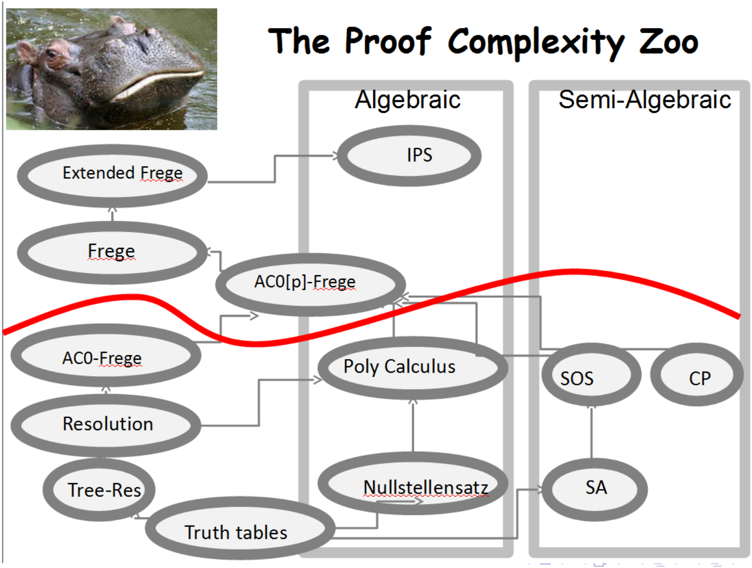
$\bar{d} \vee b$

size = 11

width = 3



Relations between proof systems



Automatizability

Automatizability [Bonet-Pitassi-Raz]

A proof system $\mathcal{P}$ is automatizable if there exists an algorithm $A : \text{UNSAT} \to \mathcal{P}$ that takes as input τ and returns a $\mathcal{P}$-refutation of τ in time poly(n, S), where $S := S_{\mathcal{P}}(\tau)$.

Automatizability

Automatizability [Bonet-Pitassi-Raz]

A proof system $\mathcal{P}$ is f-automatizable if there exists an algorithm $A : \text{UNSAT} \to \mathcal{P}$ that takes as input τ and returns a $\mathcal{P}$-refutation of τ in time $f(n, S)$, where $S := S_{\mathcal{P}}(\tau)$.

Automatizability

Automatizability [Bonet-Pitassi-Raz]

A proof system $\mathcal{P}$ is f-automatizable if there exists an algorithm A : UNSAT $\rightarrow \mathcal{P}$ that takes as input τ and returns a $\mathcal{P}$-refutation of τ in time $f(n, S)$, where $S := S_{\mathcal{P}}(\tau)$.

Automatizability is connnected to many problems in computer science...

- theorem proving and SAT solvers ([Davis-Putnam-Logemann-Loveland], [Pipatsrisawat-Darwiche])
- algorithms for PAC learning ([Kothari-Livni], [Alekhnovich-Braverman-Feldman-Klivans-Pitassi])
- algorithms for unsupervised learning ([Bhattiprolu-Guruswami-Lee])
- approximation algorithms (many works...)

Known automatizability results

- any polynomially bounded PPS is not automatizable if NP $\not\subseteq$ P/poly ([Ajtai]; [Impagliazzo],[BPR])

Known automatizability results

- any polynomially bounded PPS is not automatizable if NP $\not\subseteq$ P/poly ([Ajtai]; [Impagliazzo],[BPR])
- approximating $S_{\mathcal{P}}(\tau)$ to within $2^{\log^{1-o(1)} n}$ is NP-hard ([Alekhnovich-Buss-Moran-Pitassi])

Known automatizability results

- any polynomially bounded PPS is not automatizable if NP $\not\subseteq$ P/poly ([Ajtai]; [Impagliazzo],[BPR])
- approximating $S_{\mathcal{P}}(\tau)$ to within $2^{\log^{1-o(1)} n}$ is NP-hard ([Alekhnovich-Buss-Moran-Pitassi])
- lower bounds against strong (Frege/Extended Frege) systems under cryptographic assumptions ([Bonet-Domingo-Gavaldà-Maciel-Pitassi],[BPR],[Krajíček-Pudlák])

Known automatizability results

- first lower bounds against automatizability for Res, TreeRes by [Alekhnovich-Razborov]

Known automatizability results

- first lower bounds against automatizability for Res, TreeRes by [Alekhnovich-Razborov]
- extended to Nullsatz, PC by [Galesi-Lauria]

Known automatizability results

- first lower bounds against automatizability for Res, TreeRes by [Alekhnovich-Razborov]
- extended to Nullsatz, PC by [Galesi-Lauria]

Rest of this talk: a new version of [AR] + [GL]

- simplified
- stronger lower bounds (near quasipolynomial)
- works for more systems (Res, TreeRes, Nullsatz, PC, Res(k))

Our results

Theorem (Main Theorem for GapETH)

Assuming GapETH, $\mathcal{P}$ *is not* $n^{\tilde{o}(\log \log S)}$*-automatizable for* $\mathcal{P} =$ Res, TreeRes, Nullsatz, PC.

Theorem (Main Theorem for ETH)

Assuming ETH, $\mathcal{P}$ *is not* $n^{\tilde{o}(\log^{1/7-o(1)} \log S)}$*-automatizable for* $\mathcal{P} =$ Res, TreeRes, Nullsatz, PC.

Our results

Theorem (Main Theorem for GapETH)

Assuming GapETH, $\mathcal{P}$ *is not* $n^{\tilde{o}(\log\log S)}$*-automatizable for* $\mathcal{P} =$ Res, TreeRes, Nullsatz, PC.

Known automatizability results

System	Assumption	Result	Ref
Any PPS	NP-hard	$2^{\log^{1-o(1)} n}$	[ABMP]
Any poly PPS	NP $\not\subseteq$ P/poly	superpoly(n, S)	[A]; [I],[BPR]
AC^0-Frege	Diffie-Hellman requires circuits of size 2^{n^ϵ}	superpoly(n, S)	[BDGMP]
Frege	Factoring Blum integers requires circuits of size $n^{\omega(1)}$	superpoly(n, S)	[BPR]
E. Frege	Discrete log is not in P/poly	superpoly(n, S)	[KP]
Res, TreeRes	W[P] $\neq$ FPT	superpoly(n, S)	[AR]
Nullsatz, PC	W[P] $\neq$ FPT	superpoly(n, S)	[GL]
Res, TreeRes, Nullsatz, PC	GapETH ETH	$n^{\tilde{\Omega}(\log \log S)}$ $n^{\tilde{\Omega}(\log^{1/7-o(1)} \log S)}$	this work

A note on width automatizability

Theorem (Observation)

If τ has a width d TreeRes or Res refutation, it can be found in time $n^{O(d)}$.

Proof: brute force (repeatedly resolve all pairs of available clauses)

A note on width automatizability

Theorem (Clegg-Edmonds-Impagliazzo)

If τ has a degree d Nullsatz *or* PC *refutation, it can be found in time $n^{O(d)}$.*

Proof: Groebner basis algorithm

A note on width automatizability

Theorem (Sherali-Adams; Shor, Parrilo-Lasserre)

If τ has a degree d SA or SoS refutation, it can be found in time $n^{O(d)}$.

Proof: linear/semidefinite programming

A note on width automatizability

Theorem (BP; CEI; SA; S, PL)

If τ has a width d TreeRes or Res refutation, it can be found in time $n^{O(d)}$. If τ has a degree d Nullsatz, PC, SA, or SoS refutation, it can be found in time $n^{O(d)}$.

Theorem (Bonet-Galesi; Lauria-Nordström, Atserias-Lauria-Nordström)

There exist τ such that $w_{\mathcal{P}}(\tau) = O(d)$ and $S_{\mathcal{P}}(\tau) = n^{\Omega(d)}$ for $\mathcal{P} =$ TreeRes, Res.
There exist τ such that $deg_{\mathcal{P}}(\tau) = O(d)$ and $S_{\mathcal{P}}(\tau) = n^{\Omega(d)}$ for $\mathcal{P} =$ Nullsatz, PC, SA, SoS.

A note on width automatizability

Theorem (BP; CEI; SA; S, PL)

If τ has a width d TreeRes *or* Res *refutation, it can be found in time $n^{O(d)}$. If τ has a degree d* Nullsatz, PC, SA, *or* SoS *refutation, it can be found in time $n^{O(d)}$.*

Theorem (Bonet-Galesi; Lauria-Nordström, Atserias-Lauria-Nordström)

There exist τ such that $w_{\mathcal{P}}(\tau) = O(d)$ and $S_{\mathcal{P}}(\tau) = n^{\Omega(d)}$ for $\mathcal{P} =$ TreeRes, Res.
There exist τ such that $deg_{\mathcal{P}}(\tau) = O(d)$ and $S_{\mathcal{P}}(\tau) = n^{\Omega(d)}$ for $\mathcal{P} =$ Nullsatz, PC, SA, SoS.

Important: does *not* mean that automatizability is resolved, because $S_{\mathcal{P}} = n^{O(d)}$ may not be tight.

A note on width automatizability

Theorem (Ben-Sasson-Wigderson)

$w(\tau) \leq \log S(\tau)$ *for* TreeRes *and* $w(\tau) \leq \sqrt{n \log S(\tau)}$ *for* Res.

A note on width automatizability

Theorem (Ben-Sasson-Wigderson)

$w(\tau) \leq \log S(\tau)$ *for* TreeRes *and* $w(\tau) \leq \sqrt{n \log S(\tau)}$ *for* Res.

Theorem (BP)

TreeRes *is* $n^{O(\log S)}$*-automatizable.*
Res *is* $n^{O(\sqrt{n \log S})}$*-automatizable.*

A note on width automatizability

Theorem (Ben-Sasson-Wigderson)

$w(\tau) \leq \log S(\tau)$ *for* TreeRes *and* $w(\tau) \leq \sqrt{n \log S(\tau)}$ *for* Res.

Theorem (BP)

TreeRes *is* $n^{O(\log S)}$*-automatizable.*
Res *is* $n^{O(\sqrt{n \log S})}$*-automatizable.*

Nullsatz is $n^{O(\log S)}$-automatizable, no other upper bounds known.

Getting an automatizability lower bound

Recipe:

(1) Hard gap problem G

(2) Turn an instance of G into a tautology τ such that

- "yes" instances have small proofs
- "no" instances have no small proofs

(3) Run automatizing algorithm *Aut* on τ and see how long the output is

Getting an automatizability lower bound

Recipe:

(1) Hard gap problem G

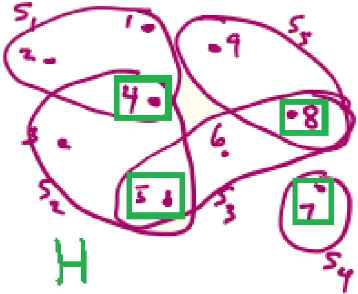
(2) Turn an instance of G into a tautology τ such that

- "yes" instances have small proofs
- "no" instances have no small proofs

(3) Run automatizing algorithm Aut on τ and see how long the output is

Gap hitting set

- $\mathcal{S} = \{S_1 \dots S_n\}$ over $[n]$
- *hitting set*: $H \subseteq [n]$ s.t. $H \cap S_i \neq \emptyset$ for all $i \in [n]$
- $\gamma(\mathcal{S})$ is the size of the smallest such H
- **Gap hitting set**: given $\mathcal{S}$, distinguish whether $\gamma(\mathcal{S}) \leq k$ or $\gamma(\mathcal{S}) > k^2$



Theorem (CCKLMNT)

Assuming GapETH *the gap hitting set problem cannot be solved in time* $n^{o(k)}$ *for* $k = \tilde{O}(\log \log n)$

Getting an automatizability lower bound

Recipe:

(1) Hard gap problem G

(2) Turn an instance of G into a tautology τ such that

- "yes" instances have small proofs
- "no" instances have no small proofs

(3) Run automatizing algorithm Aut on τ and see how long the output is

From gap hitting set to automatizability

Theorem (Main Technical Lemma)

For $k = \tilde{O}(\log\log n)$, *there exists a polytime algorithm mapping* $\mathcal{S}$ *to* $\tau_{\mathcal{S}}$ *s.t.*

- *if* $\gamma(\mathcal{S}) \leq k$ *then* $S_{\mathcal{P}}(\tau_{\mathcal{S}}) \leq n^{O(1)}$
- *if* $\gamma(\mathcal{S}) > k^2$ *then* $S_{\mathcal{P}}(\tau_{\mathcal{S}}) \geq n^{\Omega(k)}$

where $\mathcal{P} \in \{\text{TreeRes}, \text{Res}, \text{Nullsatz}, \text{PC}\}$.

Getting an automatizability lower bound

Recipe:

(1) Hard gap problem G

(2) Turn an instance of G into a tautology τ such that

- "yes" instances have small proofs
- "no" instances have no small proofs

(3) Run automatizing algorithm *Aut* on τ and see how long the output is

Proof of main theorem

Theorem (Main Theorem)

Assuming GapETH, *$\mathcal{P}$ is not $n^{\tilde{o}(\log\log S)}$-automatizable.*

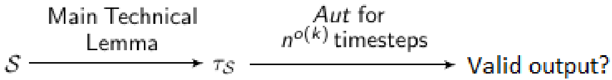
Proof: Let *Aut* be the automatizing algorithm for $\mathcal{P}$ running in time $f(n, S) = n^{\tilde{o}(\log\log S)}$, and let $k = \tilde{\Theta}(\log\log n)$.

Proof of main theorem

Theorem (Main Theorem)

Assuming GapETH, $\mathcal{P}$ *is not* $n^{\tilde{o}(\log\log S)}$*-automatizable.*

Proof: Let *Aut* be the automatizing algorithm for $\mathcal{P}$ running in time $f(n, S) = n^{\tilde{o}(\log\log S)}$, and let $k = \tilde{\Theta}(\log\log n)$.

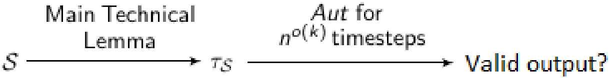


Proof of main theorem

Theorem (Main Theorem)

Assuming GapETH, *$\mathcal{P}$ is not $n^{\tilde{o}(\log\log S)}$-automatizable.*

Proof: Let *Aut* be the automatizing algorithm for $\mathcal{P}$ running in time $f(n,S) = n^{\tilde{o}(\log\log S)}$, and let $k = \tilde{\Theta}(\log\log n)$.



Theorem (Main Technical Lemma)

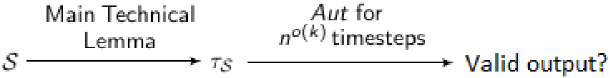
- *if* $\gamma(\mathcal{S}) \leq k$ *then* $S \leq n^{O(1)}$
- *if* $\gamma(\mathcal{S}) > k^2$ *then* $S \geq n^{\Omega(k)}$

Proof of main theorem

Theorem (Main Theorem)

Assuming GapETH, $\mathcal{P}$ *is not* $n^{\tilde{o}(\log \log S)}$*-automatizable.*

Proof: Let *Aut* be the automatizing algorithm for $\mathcal{P}$ running in time $f(n, S) = n^{\tilde{o}(\log \log n)} = n^{o(k)}$, and let $k = \tilde{\Theta}(\log \log n)$.



Theorem (Main Technical Lemma)

- *if* $\gamma(\mathcal{S}) \leq k$ *then* $S \leq n^{O(1)}$
- *if* $\gamma(\mathcal{S}) > k^2$ *then* $S \geq n^{\Omega(k)}$

Proof of main theorem

Theorem (Main Theorem)

Assuming GapETH, $\mathcal{P}$ *is not* $n^{\tilde{o}(\log\log S)}$*-automatizable.*

Proof: Let *Aut* be the automatizing algorithm for $\mathcal{P}$ running in time $f(n, S) = n^{\tilde{o}(\log\log S)}$, and let $k = \tilde{\Theta}(\log\log n)$.

$\mathcal{S}$ —(Main Technical Lemma)→ $\tau_{\mathcal{S}}$ —(*Aut* for $n^{o(k)}$ timesteps)→ Valid output?

Theorem (Main Technical Lemma)

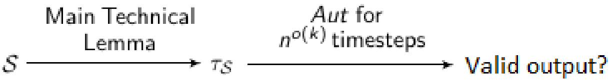
- *if* $\gamma(\mathcal{S}) \leq k$ *then* $S \leq n^{O(1)}$
- *if* $\gamma(\mathcal{S}) > k^2$ *then* $S \geq n^{\Omega(k)}$

Proof of main theorem

Theorem (Main Theorem)

Assuming GapETH, $\mathcal{P}$ *is not* $n^{\tilde{o}(\log\log S)}$*-automatizable.*

Proof: Let *Aut* be the automatizing algorithm for $\mathcal{P}$ running in time $f(n,S) = n^{\tilde{o}(\log\log S)}$, and let $k = \tilde{\Theta}(\log\log n)$.



Theorem (CCKLMNT)

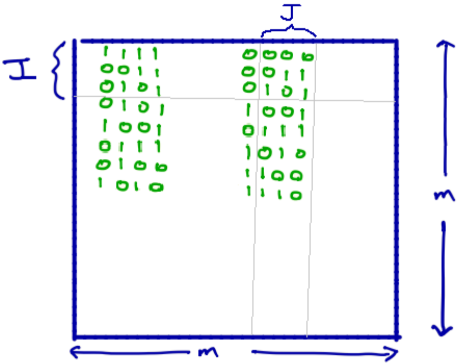
Assuming GapETH *the gap hitting set problem cannot be solved in time* $n^{o(k)}$ *for* $k = \tilde{O}(\log\log n)$

For the rest of the talk...

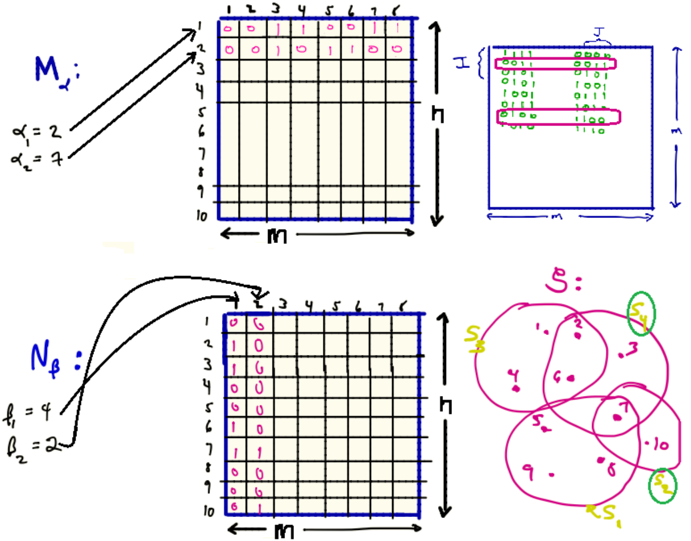
- fix $k = \tilde{\Theta}(\log \log n)$
- $m = n^{1/k}$ $(k \log m = \log n)$
- $k \leq \frac{\log m}{4}$

Detour: universal sets

- $A_{m \times m}$ is (m, q)-*universal* if for all $I \subseteq [m]$, $|I| \leq q$, all $2^{|I|}$ possible column vectors appear in A restricted to the rows I
- additional requirement: for all $J \subseteq [m]$, $|J| \leq q$, all $2^{|J|}$ possible row vectors appear in A restricted to the columns J
- fix some such A as a gadget (constructions like the *Paley graph* work for $q = \frac{\log m}{4}$)



Defining τ_S



Defining $\tau_{\mathcal{S}}$

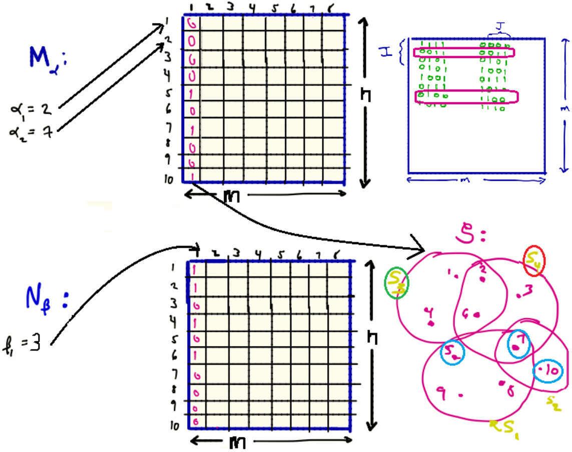
- $Mat(\mathcal{S})_{n \times n}$ is the matrix whose columns are the indicator vectors of $\mathcal{S}$
- $\vec{x} = x_1 \ldots x_n$ where $x_i \in \{0,1\}^{\log m}$ ($n \log m$ variables total),
  $\vec{y} = y_1 \ldots y_m$ where $y_j \in \{0,1\}^{\log n}$ ($m \log n$ variables total)
- $x_i = \alpha_i \to M_\alpha[i,j] = A[\alpha_i, j]$ (treat α_i as an element of $[m]$)
- $y_j = \beta_j \to N_\beta[i,j] = Mat(\mathcal{S})[i, \beta_j]$ (treat β_j as an element of $[n]$)

Defining $\tau_{\mathcal{S}}$

$\tau_{\mathcal{S}}$ will state that there exist $\vec{\alpha}, \vec{\beta}$ such that there is no i, j where $M_\alpha[i,j] = N_\beta[i,j] = 1$

Defining $\tau_{\mathcal{S}}$

$\tau_{\mathcal{S}}$ will state that there exist $\vec{\alpha}, \vec{\beta}$ such that there is no i, j where $M_\alpha[i,j] = N_\beta[i,j] = 1$



Defining $\tau_{\mathcal{S}}$

$\tau_{\mathcal{S}}$ will state that there exist $\vec{\alpha}, \vec{\beta}$ such that there is no i, j where $M_\alpha[i,j] = N_\beta[i,j] = 1$

- for every i, j, α_i, β_j such that $A[\alpha_i, j] = Mat(\mathcal{S})[i, \beta_j] = 1$,

$$\overline{x_i^{\alpha_i} \wedge y_j^{\beta_j}}$$

- all clauses have width $\log m + \log n$
- $nm2^{\log n}2^{\log m} = n^2m^2$ clauses

Defining $\tau_{\mathcal{S}}$

$\tau_{\mathcal{S}}$ will state that there exist $\vec{\alpha}, \vec{\beta}$ such that there is no i, j where $M_\alpha[i,j] = N_\beta[i,j] = 1$

- for every i, j, α_i, β_j such that $A[\alpha_i, j] = Mat(\mathcal{S})[i, \beta_j] = 1$,

$$\overline{x_i^{\alpha_i} \wedge y_j^{\beta_j}}$$

- all clauses have width $\log m + \log n$
- $nm2^{\log n}2^{\log m} = n^2m^2$ clauses

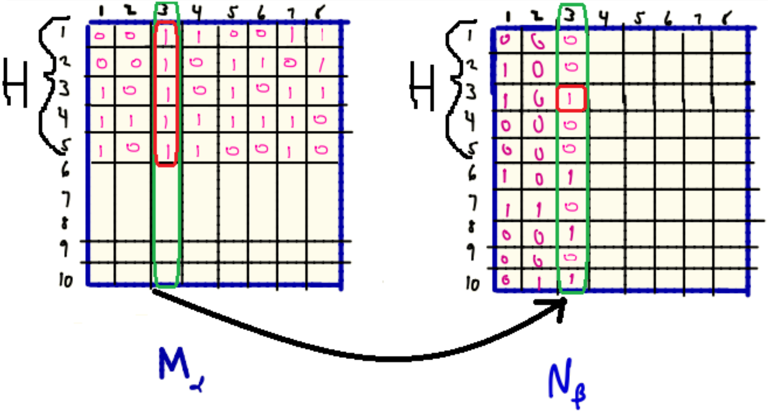
Lemma

$\tau_{\mathcal{S}}$ is unsatisfiable when $\gamma(\mathcal{S}) \leq \frac{\log m}{4}$.

Defining $\tau_{\mathcal{S}}$

Lemma

$\tau_{\mathcal{S}}$ is unsatisfiable when $\gamma(\mathcal{S}) \leq \frac{\log m}{4}$.



Defining $\tau_{\mathcal{S}}$

Lemma

$\tau_{\mathcal{S}}$ is unsatisfiable when $\gamma(\mathcal{S}) \leq \frac{\log m}{4}$.

Proof: Let $H = \{i_1 \ldots i_\gamma\}$ be a hitting set of size $\gamma := \gamma(\mathcal{S})$.

Defining $\tau_{\mathcal{S}}$

Lemma

$\tau_{\mathcal{S}}$ is unsatisfiable when $\gamma(\mathcal{S}) \leq \frac{\log m}{4}$.

Proof: Let $H = \{i_1 \dots i_\gamma\}$ be a hitting set of size $\gamma := \gamma(\mathcal{S})$.
$\{\alpha_{i_1} \dots \alpha_{i_\gamma}\}$ is a set of at most $\frac{\log m}{4}$ rows from A ($\gamma \leq \frac{\log m}{4}$).

Defining $\tau_{\mathcal{S}}$

Lemma

$\tau_{\mathcal{S}}$ is unsatisfiable when $\gamma(\mathcal{S}) \leq \frac{\log m}{4}$.

Proof: Let $H = \{i_1 \dots i_\gamma\}$ be a hitting set of size $\gamma := \gamma(\mathcal{S})$.
$\{\alpha_{i_1} \dots \alpha_{i_\gamma}\}$ is a set of at most $\frac{\log m}{4}$ rows from A ($\gamma \leq \frac{\log m}{4}$).
There exists some $j \in [m]$ such that $M_\alpha[i, j] = 1$ for all $i \in H$ (universal property of A).

Defining $\tau_{\mathcal{S}}$

Lemma

$\tau_{\mathcal{S}}$ is unsatisfiable when $\gamma(\mathcal{S}) \leq \frac{\log m}{4}$.

Proof: Let $H = \{i_1 \ldots i_\gamma\}$ be a hitting set of size $\gamma := \gamma(\mathcal{S})$.
$\{\alpha_{i_1} \ldots \alpha_{i_\gamma}\}$ is a set of at most $\frac{\log m}{4}$ rows from A ($\gamma \leq \frac{\log m}{4}$).
There exists some $j \in [m]$ such that $M_\alpha[i,j] = 1$ for all $i \in H$ (universal property of A).
There must be some $i \in H$ such that $N_\beta[i,j] = 1$ (H is a hitting set).

Defining $\tau_{\mathcal{S}}$

Lemma

$\tau_{\mathcal{S}}$ is unsatisfiable when $\gamma(\mathcal{S}) \leq \frac{\log m}{4}$.

Proof: Let $H = \{i_1 \dots i_\gamma\}$ be a hitting set of size $\gamma := \gamma(\mathcal{S})$.
$\{\alpha_{i_1} \dots \alpha_{i_\gamma}\}$ is a set of at most $\frac{\log m}{4}$ rows from A ($\gamma \leq \frac{\log m}{4}$).
There exists some $j \in [m]$ such that $M_\alpha[i,j] = 1$ for all $i \in H$ (universal property of A).
There must be some $i \in H$ such that $N_\beta[i,j] = 1$ (H is a hitting set).
Therefore the axiom $\overline{x_i^{\alpha_i} \wedge y_j^{\beta_j}}$ is falsified.

Upper bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Upper bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$)

If $\gamma(\mathcal{S}) \leq k$, then $S_{\mathcal{P}}(\tau_{\mathcal{S}}) \leq n^{O(1)}$ for any $\mathcal{P}$ which p-simulates TreeRes.

Upper bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Upper bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$)

If $\gamma(\mathcal{S}) \leq k$, then $S_{\mathcal{P}}(\tau_{\mathcal{S}}) \leq n^{O(1)}$ for any $\mathcal{P}$ which p-simulates TreeRes.

Proof: TreeRes refutation of τ $\leftrightarrow$ decision tree solving the search problem on τ

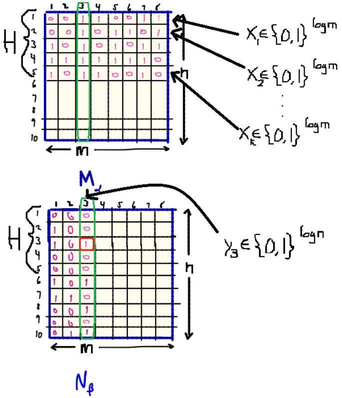
Upper bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Upper bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$)

If $\gamma(\mathcal{S}) \leq k$, then $S_{\mathcal{P}}(\tau_{\mathcal{S}}) \leq n^{O(1)}$ for any $\mathcal{P}$ which p-simulates TreeRes.

Proof: TreeRes refutation of $\tau \leftrightarrow$ decision tree solving the search problem on τ

- query all vars in x_i for all $i \in H$
- find the j with all 1s
- query all vars in y_j



Upper bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

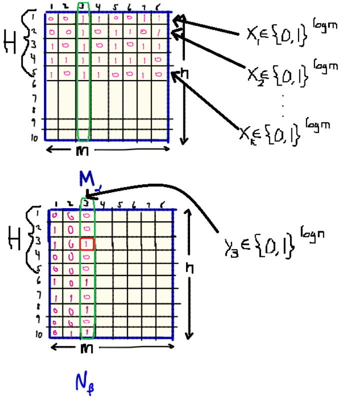
Lemma (Upper bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$)

If $\gamma(\mathcal{S}) \leq k$, then $S_{\mathcal{P}}(\tau_{\mathcal{S}}) \leq n^{O(1)}$ for any $\mathcal{P}$ which p-simulates TreeRes.

Proof: TreeRes refutation of $\tau \leftrightarrow$ decision tree solving the search problem on τ

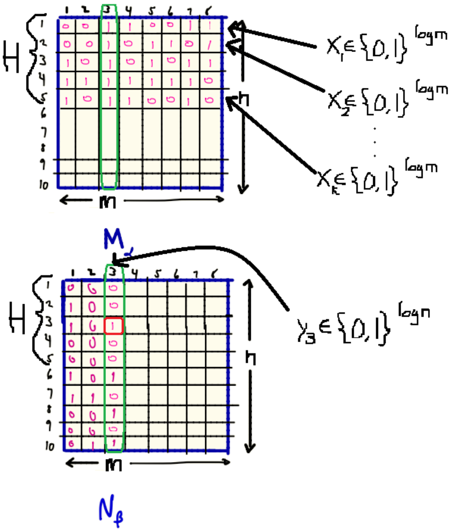
- query all vars in x_i for all $i \in H$
- find the j with all 1s
- query all vars in y_j

Size of the proof:
$2^{k \log m + \log n} = n^2$



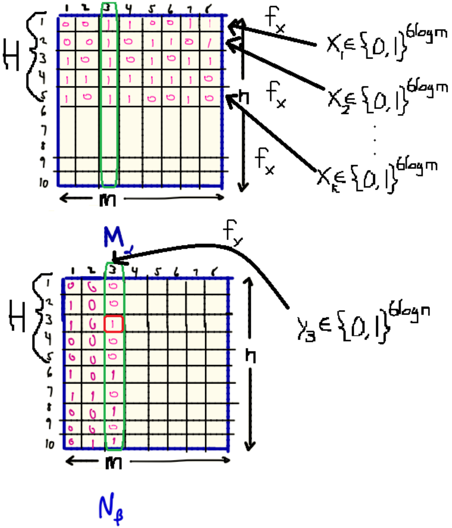
Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

- *error-correcting codes*:
  $x_i \in \{0,1\}^{6 \log m}$,
  $y_j \in \{0,1\}^{6 \log n}$



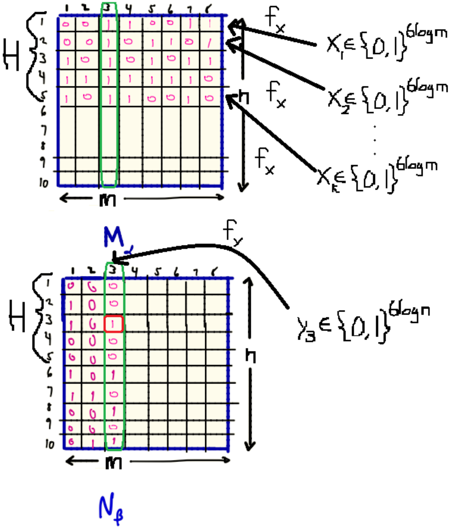
Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

- *error-correcting codes*: $x_i \in \{0,1\}^{6 \log m}$, $y_j \in \{0,1\}^{6 \log n}$
- $f_x : \{0,1\}^{6 \log m} \to \{0,1\}^{\log m}$ is $2 \log m$*-surjective*, $f_y : \{0,1\}^{6 \log n} \to \{0,1\}^{\log n}$ is $2 \log$ *n-surjective*



Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

- *error-correcting codes*:
  $x_i \in \{0,1\}^{6 \log m}$,
  $y_j \in \{0,1\}^{6 \log n}$
- $f_x : \{0,1\}^{6 \log m} \to \{0,1\}^{\log m}$ is $2 \log m$*-surjective*,
  $f_y : \{0,1\}^{6 \log n} \to \{0,1\}^{\log n}$ is $2 \log n$*-surjective*
- high-level idea: π knows nothing about a row or column without setting lots of variables



Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

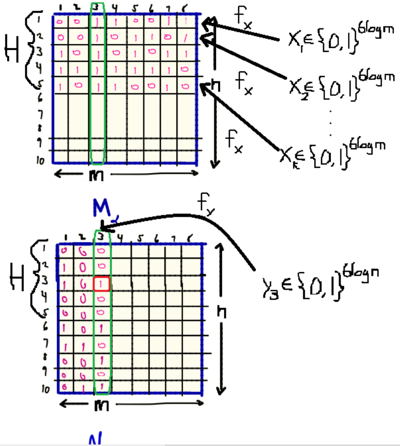
Lemma (Upper bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$)

If $\gamma(\mathcal{S}) \leq k$, then $S_{\mathcal{P}}(\tau_{\mathcal{S}}) \leq n^{O(1)}$ for any $\mathcal{P}$ which p-simulates TreeRes.

Proof: TreeRes refutation of $\tau \leftrightarrow$ decision tree solving the search problem on τ

- query all vars in x_i for all $i \in H$
- find the j with all 1s
- query all vars in y_j

Size of the proof:
$2^{6k \log m + 6 \log n} = n^{12}$



Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Lower bound on $S(\tau_{\mathcal{S}})$)

If $\gamma(\mathcal{S}) > k^2$, then $S_{\mathcal{P}}(\tau_{\mathcal{S}}) \geq n^{\Omega(k)}$.

Two steps:

1. Width/degree lower bound
2. Random restriction argument

Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Lower bound on $S(\tau_{\mathcal{S}})$ for TreeRes)

If $\gamma(\mathcal{S}) > k^2$, then $S_{\mathcal{P}}(\tau_{\mathcal{S}}) \geq n^{\Omega(k)}$ for $\mathcal{P} =$ TreeRes.

One step:

1. Height lower bound

Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

To get height lower bounds, we play an adversarial game against π solving the search problem.

Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

To get height lower bounds, we play an adversarial game against π solving the search problem.

- path p in a TreeRes refutation π is a partial restriction to $\tau_{\mathcal{S}}$
- $I_0(p) = \{i \in [n] \mid p \text{ contains at least } \log m \text{ literals from } x_i\}$
- $J_0(p) = \{j \in [m] \mid p \text{ contains at least } \log n \text{ literals from } y_j\}$

Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

To get height lower bounds, we play an adversarial game against π solving the search problem.

- path p in a TreeRes refutation π is a partial restriction to $\tau_{\mathcal{S}}$
- $I_0(p) = \{i \in [n] \mid p \text{ contains at least } \log m \text{ literals from } x_i\}$
- $J_0(p) = \{j \in [m] \mid p \text{ contains at least } \log n \text{ literals from } y_j\}$

Lemma (Row/column height lower bound for TreeRes)

If $\gamma(\mathcal{S}) > k^2$, then for every TreeRes *refutation π for $\tau_{\mathcal{S}}$, π contains a path p such that either $|I_0(p)| \geq k^2$ or $|J_0(p)| \geq k$.*

Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

To get height lower bounds, we play an adversarial game against π solving the search problem.

- path p in a TreeRes refutation π is a partial restriction to $\tau_{\mathcal{S}}$
- $I_0(p) = \{i \in [n] \mid p \text{ contains at least } \log m \text{ literals from } x_i\}$
- $J_0(p) = \{j \in [m] \mid p \text{ contains at least } \log n \text{ literals from } y_j\}$

Lemma (Row/column height lower bound for TreeRes)

If $\gamma(\mathcal{S}) > k^2$, then for every TreeRes *refutation π for $\tau_{\mathcal{S}}$, π contains a path p such that either $|I_0(p)| \geq k^2$ or $|J_0(p)| \geq k$.*

Corollary (Height lower bound for TreeRes)

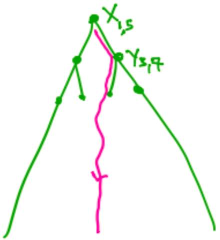
If $\gamma(\mathcal{S}) > k^2$, then for every TreeRes *refutation π for $\tau_{\mathcal{S}}$, π has height at least $k \log n$.*

Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Row/column height lower bound for TreeRes)

If $\gamma(\mathcal{S}) > k^2$, then for every TreeRes *refutation π for $\tau_{\mathcal{S}}$, π contains a path p such that either $|I_0(p)| \geq k^2$ or $|J_0(p)| \geq k$.*

Proof: We play an adversarial game against π solving the search problem.



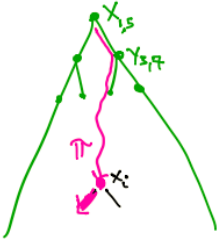
Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Row/column height lower bound for TreeRes)

If $\gamma(\mathcal{S}) > k^2$, then for every TreeRes *refutation π for $\tau_{\mathcal{S}}$, π contains a path p such that either $|I_0(p)| \geq k^2$ or $|J_0(p)| \geq k$.*

Whenever π queries a variable in x_i:

- if p contains less than $\log m$ x_i variables ($i \notin I_0(p)$) we branch arbitrarily



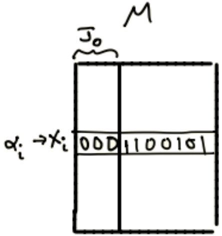
Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Row/column height lower bound for TreeRes)

If $\gamma(\mathcal{S}) > k^2$, then for every TreeRes refutation π for $\tau_{\mathcal{S}}$, π contains a path p such that either $|I_0(p)| \geq k^2$ or $|J_0(p)| \geq k$.

Whenever π queries a variable in x_i:

- if this is the $\log m$th variable in x_i, we choose some $a_i \in A$ such that $(a_i)_j = 0$ for all $j \in J_0(p)$ ($|J_0(p)| < k \leq \frac{\log m}{4}$)



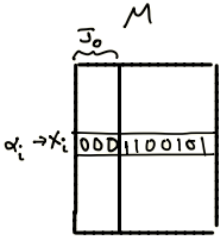
Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Row/column height lower bound for TreeRes)

If $\gamma(\mathcal{S}) > k^2$, then for every TreeRes refutation π for $\tau_{\mathcal{S}}$, π contains a path p such that either $|I_0(p)| \geq k^2$ or $|J_0(p)| \geq k$.

Whenever π queries a variable in x_i:

- if this is the $\log m$th variable in x_i, we choose some $a_i \in A$ such that $(a_i)_j = 0$ for all $j \in J_0(p)$ ($|J_0(p)| < k \leq \frac{\log m}{4}$) and some assignment α_i consistent with p such that $f_x(\alpha_i) = a_i$ (p has only queried $\log m$ variables in x_i so far).



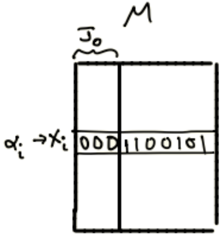
Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Row/column height lower bound for TreeRes)

If $\gamma(\mathcal{S}) > k^2$, then for every TreeRes refutation π for $\tau_{\mathcal{S}}$, π contains a path p such that either $|I_0(p)| \geq k^2$ or $|J_0(p)| \geq k$.

Whenever π queries a variable in x_i:

- if this is the $\log m$th variable in x_i, we choose some $a_i \in A$ such that $(a_i)_j = 0$ for all $j \in J_0(p)$ ($|J_0(p)| < k \leq \frac{\log m}{4}$) and some assignment α_i consistent with p such that $f_x(\alpha_i) = a_i$ (p has only queried $\log m$ variables in x_i so far). Store α_i and add i to $I_0(p)$.



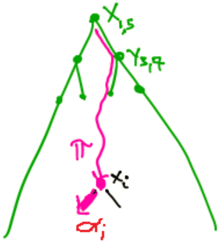
Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Row/column height lower bound for TreeRes)

If $\gamma(\mathcal{S}) > k^2$, then for every TreeRes *refutation π for $\tau_{\mathcal{S}}$, π contains a path p such that either $|I_0(p)| \geq k^2$ or $|J_0(p)| \geq k$.*

Whenever π queries a variable in x_i:

- if $i \in I_0(p)$ we answer according to the stored α_i



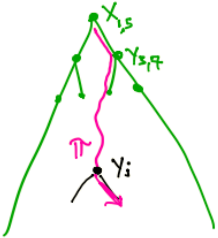
Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Row/column height lower bound for TreeRes)

If $\gamma(\mathcal{S}) > k^2$, then for every TreeRes *refutation π for $\tau_{\mathcal{S}}$, π contains a path p such that either $|I_0(p)| \geq k^2$ or $|J_0(p)| \geq k$.*

Whenever π queries a variable in y_j:

- if p contains less than $\log n$ y_j variables ($j \notin J_0(p)$) we branch arbitrarily



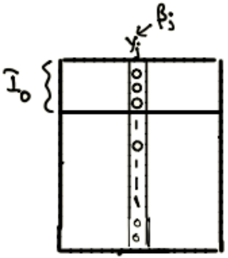
Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Row/column height lower bound for TreeRes)

If $\gamma(\mathcal{S}) > k^2$, then for every TreeRes refutation π for $\tau_{\mathcal{S}}$, π contains a path p such that either $|I_0(p)| \geq k^2$ or $|J_0(p)| \geq k$.

Whenever π queries a variable in y_j:

- if this is the $\log n$th variable in y_j, we choose some $S_j \in Mat(\mathcal{S})$ such that $(S_j)_i = 0$ for all $i \in I_0(p)$ $(|I_0(p)| < k^2 < \gamma(\mathcal{S}))$



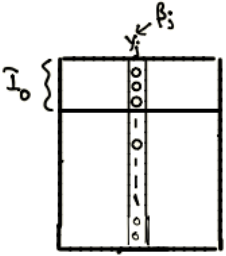
Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Row/column height lower bound for TreeRes)

If $\gamma(\mathcal{S}) > k^2$, then for every TreeRes *refutation π for $\tau_{\mathcal{S}}$, π contains a path p such that either $|I_0(p)| \geq k^2$ or $|J_0(p)| \geq k$.*

Whenever π queries a variable in y_j:

- if this is the $\log n$th variable in y_j, we choose some $S_j \in Mat(\mathcal{S})$ such that $(S_j)_i = 0$ for all $i \in I_0(p)$ ($|I_0(p)| < k^2 < \gamma(\mathcal{S})$) and some assignment β_j consistent with p such that $f_y(\beta_j) = S_j$ (p has only queried $\log n$ variables in y_j so far).



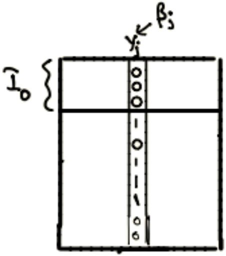
Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Row/column height lower bound for TreeRes)

If $\gamma(\mathcal{S}) > k^2$, then for every TreeRes refutation π for $\tau_{\mathcal{S}}$, π contains a path p such that either $|I_0(p)| \geq k^2$ or $|J_0(p)| \geq k$.

Whenever π queries a variable in y_j:

- if this is the $\log n$th variable in y_j, we choose some $S_j \in Mat(\mathcal{S})$ such that $(S_j)_i = 0$ for all $i \in I_0(p)$ ($|I_0(p)| < k^2 < \gamma(\mathcal{S})$) and some assignment β_j consistent with p such that $f_y(\beta_j) = S_j$ (p has only queried $\log n$ variables in y_j so far). Store β_j and add j to $J_0(p)$.



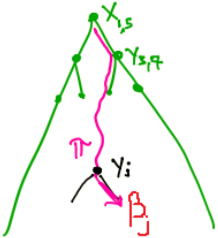
Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Row/column height lower bound for TreeRes)

If $\gamma(\mathcal{S}) > k^2$, then for every TreeRes *refutation π for $\tau_{\mathcal{S}}$, π contains a path p such that either $|I_0(p)| \geq k^2$ or $|J_0(p)| \geq k$.*

Whenever π queries a variable in y_j:

- if $j \in J_0(p)$ we answer according to the stored β_j



Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Lower bound on $S(\tau_{\mathcal{S}})$ for Res)

If $\gamma(\mathcal{S}) > k^2$, then $S_{\mathcal{P}}(\tau_{\mathcal{S}}) \geq n^{\Omega(k)}$ for $\mathcal{P} = \mathrm{Res}$.

Two steps:

1. Width lower bound
2. Random restriction argument

Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Wide clause lemma for Res)

If $\gamma(\mathcal{S}) \geq k^2$, *then for every* Res *refutation* π *for* $\tau_{\mathcal{S}}$, π *contains a clause* D *such that either* $|I_0(D)| \geq k^2$ *or* $|J_0(D)| \geq k$.

Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Wide clause lemma for Res)

If $\gamma(\mathcal{S}) \geq k^2$, then for every Res *refutation π for $\tau_{\mathcal{S}}$, π contains a clause D such that either $|I_0(D)| \geq k^2$ or $|J_0(D)| \geq k$.*

Proof: To get a width lower bound for Res, it suffices to do the same adversarial argument as with TreeRes height, but where p is allowed to "forget" literals.

Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Wide clause lemma for Res)

If $\gamma(\mathcal{S}) \geq k^2$*, then for every* Res *refutation* π *for* $\tau_{\mathcal{S}}$*,* π *contains a clause* D *such that either* $|I_0(D)| \geq k^2$ *or* $|J_0(D)| \geq k$*.*

Proof: To get a width lower bound for Res, it suffices to do the same adversarial argument as with TreeRes height, but where p is allowed to "forget" literals.

We play the exactly as in the TreeRes wide clause lemma, but now whenever i drops below the $\log m$ threshold we erase our stored α_i, and likewise for j.

Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Wide clause lemma for Res)

If $\gamma(\mathcal{S}) \geq k^2$, then for every Res *refutation π for $\tau_{\mathcal{S}}$, π contains a clause D such that either $|I_0(D)| \geq k^2$ or $|J_0(D)| \geq k$.*

Proof: To get a width lower bound for Res, it suffices to do the same adversarial argument as with TreeRes height, but where p is allowed to "forget" literals.

We play the exactly as in the TreeRes wide clause lemma, but now whenever i drops below the $\log m$ threshold we erase our stored α_i, and likewise for j.

To get a contradiction we consider the *last* time i was added to I_0 and j was added to J_0.

Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Lower bound on $S(\tau_{\mathcal{S}})$)

If $\gamma(\mathcal{S}) > k^2$, then $S_{\mathcal{P}}(\tau_{\mathcal{S}}) \geq n^{\Omega(k)}$ for $\mathcal{P} = \text{Res}$.

Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Lower bound on $S(\tau_{\mathcal{S}})$)

If $\gamma(\mathcal{S}) > k^2$, then $S_{\mathcal{P}}(\tau_{\mathcal{S}}) \geq n^{\Omega(k)}$ for $\mathcal{P} = \text{Res}$.

Proof: Assume for contradiction that $|\pi| \leq n^{o(k)}$.

Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Lower bound on $S(\tau_{\mathcal{S}})$)

If $\gamma(\mathcal{S}) > k^2$, then $S_{\mathcal{P}}(\tau_{\mathcal{S}}) \geq n^{\Omega(k)}$ for $\mathcal{P} = \text{Res}$.

Proof: Assume for contradiction that $|\pi| \leq n^{o(k)}$.
Hit it with a random restriction that sets $\log m$ x_i variables per i and $\log n$ y_j variables per j.

Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Lower bound on $S(\tau_{\mathcal{S}})$)

If $\gamma(\mathcal{S}) > k^2$, then $S_{\mathcal{P}}(\tau_{\mathcal{S}}) \geq n^{\Omega(k)}$ for $\mathcal{P} = \text{Res}$.

Proof: Assume for contradiction that $|\pi| \leq n^{o(k)}$.
Hit it with a random restriction that sets $\log m$ x_i variables per i and $\log n$ y_j variables per j.
By the probabilistic method there is a restriction ρ that sets every wide clause in π to 1.

Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Lemma (Lower bound on $S(\tau_{\mathcal{S}})$)

If $\gamma(\mathcal{S}) > k^2$, then $S_{\mathcal{P}}(\tau_{\mathcal{S}}) \geq n^{\Omega(k)}$ for $\mathcal{P} = \text{Res}$.

Proof: Assume for contradiction that $|\pi| \leq n^{o(k)}$.
Hit it with a random restriction that sets $\log m$ x_i variables per i and $\log n$ y_j variables per j.
By the probabilistic method there is a restriction ρ that sets every wide clause in π to 1.

Lemma (Wide clause lemma for Res)

If $\gamma(\mathcal{S}) \geq k^2$, then for every Res refutation π for $\tau_{\mathcal{S}}$, $\pi|_{\rho}$ contains a clause D such that either $|I_0(D)| \geq k^2$ or $|J_0(D)| \geq k$.

Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Other proof systems:

Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Other proof systems:

- Res - prover-delayer game [Pudlák, Atserias-Lauria-Nordström]

Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Other proof systems:

- Res - prover-delayer game [Pudlák, Atserias-Lauria-Nordström]
- Nullsatz + PC - linear operator [Galesi-Lauria]

Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

Other proof systems:

- Res - prover-delayer game [Pudlák, Atserias-Lauria-Nordström]
- Nullsatz + PC - linear operator [Galesi-Lauria]
- Res(k) - switching lemma [Buss-Impagliazzo-Segerlend]

Open problems

- extending to Sherali-Adams, Sum-of-Squares, Cutting Planes, . . .

Open problems

- extending to Sherali-Adams, Sum-of-Squares, Cutting Planes, . . .
- better hard k in gap hitting set $\rightarrow$ better non-automatizability result (up to $k = \sqrt{\log n}$)

Open problems

- extending to Sherali-Adams, Sum-of-Squares, Cutting Planes, . . .
- better hard k in gap hitting set $\rightarrow$ better non-automatizability result (up to $k = \sqrt{\log n}$)
- different technique that doesn't work for TreeRes may give subexponential lower bounds

Thank you!

~~ə:ˈtɒmətaɪzəˈbɪlɪti~~ ɔ:tɒˈmætaɪzəˈbɪlɪti