Short Proofs are Hard to Find

Ian Mertz

University of Toronto

Joint work w/ Toniann Pitassi, Hao Wei

ICALP, July 10, 2019

Proof propositional 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 $1 + 1 = 2$.

Whitehead, A. N., & Russell, B. (1925). *Principia mathematica*. Cambridge [England]: The University Press. pp.379

Proof propositional 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 $1 + 1 = 2$.

Whitehead, A. N., & Russell, B. (1925). *Principia mathematica*. Cambridge [England]: The University Press. pp.379

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

Proof propositional 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 $1 + 1 = 2$.

Whitehead, A. N., & Russell, B. (1925). *Principia mathematica*. Cambridge [England]: The University Press. pp.379

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

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

Resolution

One of the simplest and most important proof systems

Resolution

One of the simplest and most important proof systems

- SAT solvers ([Davis-Putnam-Logemann-Loveland], [Pipatsrisawat-Darwiche])
- automated theorem proving
- model checking
- planning/inference

Resolution

Axioms:

$\bar{a} \vee d$

$b \vee d$

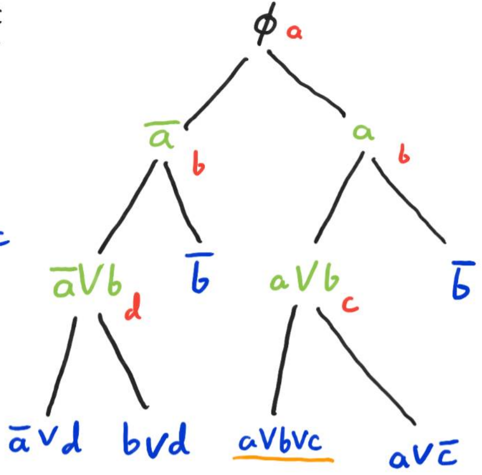
$\bar{b}$

$a \vee b \vee c$

$a \vee \bar{c}$

size = 11

width = 3



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 $f(n, S_{\mathcal{P}}(\tau))$, where $S_{\mathcal{P}}(\tau)$ is the size of the shortest $\mathcal{P}$-refutation of τ.

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 $f(n, S_{\mathcal{P}}(\tau))$, where $S_{\mathcal{P}}(\tau)$ is the size of the shortest $\mathcal{P}$-refutation of τ.

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

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

Known automatizability lower bounds

General results and results for strong systems

- approximating $S_{\mathcal{P}}(\tau)$ to within $2^{\log^{1-o(1)} n}$ is NP-hard for all "reasonable" $\mathcal{P}$ ([Alekhnovich-Buss-Moran-Pitassi])

Known automatizability lower bounds

General results and results for strong systems

- approximating $S_{\mathcal{P}}(\tau)$ to within $2^{\log^{1-o(1)} n}$ is NP-hard for all "reasonable" $\mathcal{P}$ ([Alekhnovich-Buss-Moran-Pitassi])
- lower bounds against different Frege systems under cryptographic assumptions ([Bonet-Domingo-Gavaldà-Maciel-Pitassi],[BPR],[Krajíček-Pudlák])

Known automatizability lower bounds

Results for weak systems

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

Known automatizability lower bounds

Results for weak systems

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

Known automatizability lower bounds

Results for weak systems

- 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 construction and proofs
- stronger lower bounds via ETH assumption
- results also hold for Res(r)

Our results

Theorem (Main Theorem)

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

Our results

Theorem (Main Theorem)

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

Theorem (Main Theorem for Res(r))

Assuming ETH, Res(r) *is not* $n^{\tilde{o}(\log\log S_{\mathcal{P}}(\tau)/exp(r^2))}$*-automatizable for* $r \leq \tilde{O}(\log\log\log n)$.

Our results

Theorem (Main Theorem)

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

Theorem (Atserias-Muller'19)

Assuming $\mathsf{P} \neq \mathsf{NP}$, Res *is not automatizable.*
Assuming ETH, Res *is not automatizable in subexponential time.*

Our results

Theorem (Main Theorem)

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

Theorem (Atserias-Muller'19)

Assuming P $\neq$ NP, Res *is not automatizable.*
Assuming ETH, Res *is not automatizable in subexponential time.*

Theorem (Bonet-Pitassi; Ben-Sasson-Wigderson)

TreeRes *is* $n^{O(\log S_{\mathcal{P}}(\tau))}$*-automatizable.*
Res *is* $n^{O(\sqrt{n \log S_{\mathcal{P}}(\tau)})}$*-automatizable.*

Getting an automatizability lower bound

Recipe:

(1) Hard gap problem G

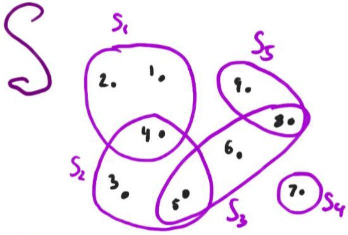
(2) Turn an instance of G into a tautology τ such that

- "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

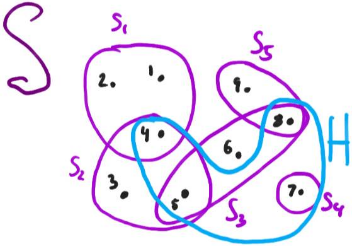
Gap hitting set

- $\mathcal{S} = \{S_1 \ldots S_n\}$ over $[n]$



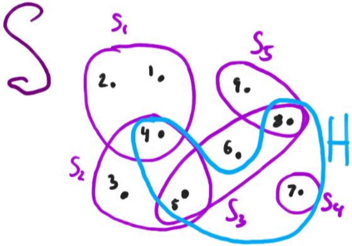
Gap hitting set

- $\mathcal{S} = \{S_1 \dots S_n\}$ over $[n]$
- *hitting set*: $H \subseteq [n]$ s.t. $H \cap S_i \neq \emptyset$ for all $i \in [n]$



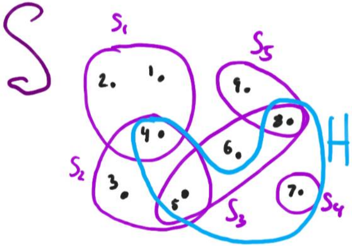
Gap hitting set

- $\mathcal{S} = \{S_1 \dots S_n\}$ 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 H
- **Gap hitting set**: given $\mathcal{S}$, distinguish whether $\gamma(\mathcal{S}) \leq k$ or $\gamma(\mathcal{S}) > k^2$



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 H
- **Gap hitting set**: given $\mathcal{S}$, distinguish whether $\gamma(\mathcal{S}) \leq k$ or $\gamma(\mathcal{S}) > k^2$



Theorem (Chen-Lin)

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

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

Proof sketch of main theorem

Theorem (Main Theorem)

Assuming ETH, *$\mathcal{P}$ is not $n^{\tilde{o}(\log\log S_{\mathcal{P}}(\tau))}$-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 sketch of main theorem

Theorem (Main Theorem)

Assuming ETH, $\mathcal{P}$ is not $n^{\tilde{o}(\log\log S_{\mathcal{P}}(\tau))}$-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 sketch of main theorem

Theorem (Main Theorem)

Assuming ETH, $\mathcal{P}$ *is not* $n^{\tilde{o}(\log\log S_{\mathcal{P}}(\tau))}$*-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* $S_{\mathcal{P}}(\tau) \leq n^{O(1)}$
- *if* $\gamma(\mathcal{S}) > k^2$ *then* $S_{\mathcal{P}}(\tau) \geq n^{\Omega(k)}$

Proof sketch of main theorem

Theorem (Main Theorem)

Assuming ETH, $\mathcal{P}$ *is not* $n^{\tilde{o}(\log\log S_{\mathcal{P}}(\tau))}$*-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* $S_{\mathcal{P}}(\tau) \leq n^{O(1)}$
- *if* $\gamma(\mathcal{S}) > k^2$ *then* $S_{\mathcal{P}}(\tau) \geq n^{\Omega(k)}$

Proof sketch of main theorem

Theorem (Main Theorem)

Assuming ETH, $\mathcal{P}$ *is not* $n^{\tilde{o}(\log\log S_{\mathcal{P}}(\tau))}$*-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)

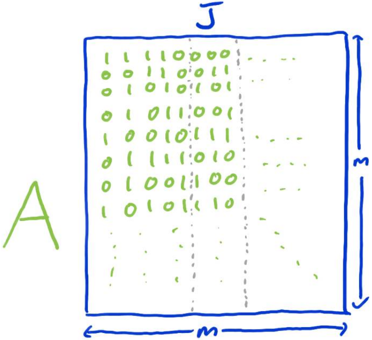
- *if* $\gamma(\mathcal{S}) \leq k$ *then* $S_{\mathcal{P}}(\tau) \leq n^{O(1)}$
- *if* $\gamma(\mathcal{S}) > k^2$ *then* $S_{\mathcal{P}}(\tau) \geq n^{\Omega(k)}$

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

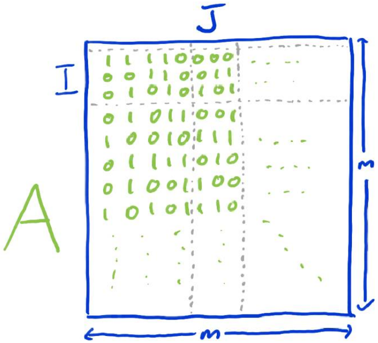
Detour: universal sets

- $A_{m \times m}$ 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
- $A_{m \times m}$ is (m, q)-*dual universal* if for all $J \subseteq [m]$, $|J| \leq q$, all $2^{|J|}$ possible row vectors appear in A restricted to the columns J



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
- $A_{m \times m}$ is (m, q)-*dual universal* if for all $J \subseteq [m]$, $|J| \leq q$, all $2^{|J|}$ possible row vectors appear in A restricted to the columns J
- constructions like the *Paley graph* work for $q = \frac{\log m}{4}$

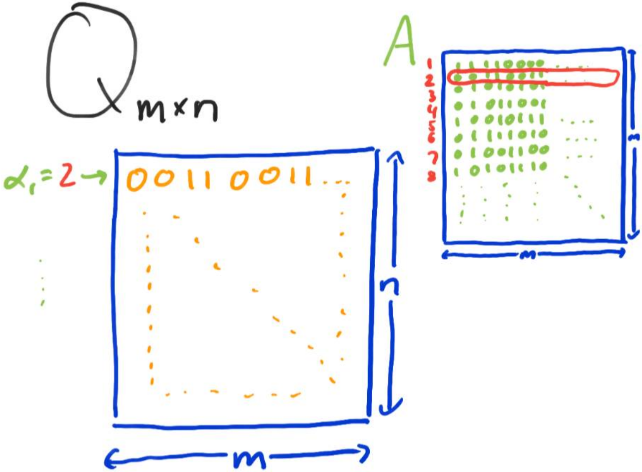


Defining $\tau_{\mathcal{S}}$

Variables of $\tau_{\mathcal{S}}$ will implicitly define two matrices using A and $\mathcal{S}$

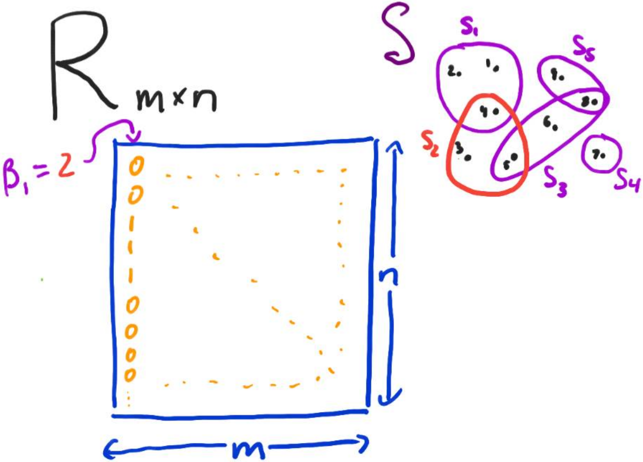
Defining $\tau_{\mathcal{S}}$

Variables of $\tau_{\mathcal{S}}$ will implicitly define two matrices using A and $\mathcal{S}$



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

Variables of $\tau_{\mathcal{S}}$ will implicitly define two matrices using A and $\mathcal{S}$

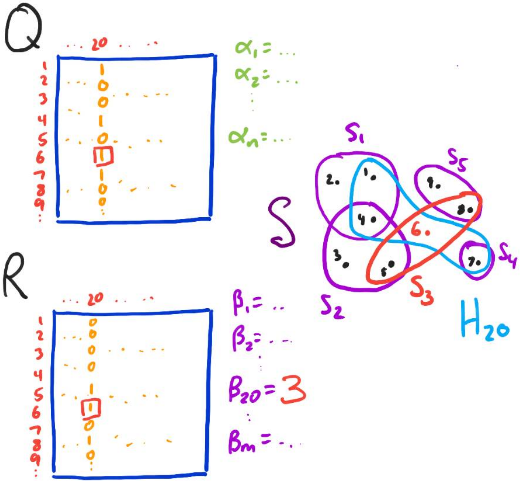


Defining $\tau_{\mathcal{S}}$

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

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

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



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

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

If $\gamma(\mathcal{S}) \leq k \leq \frac{\log m}{4}$, then $\tau_{\mathcal{S}}$ is unsatisfiable and $S(\tau_{\mathcal{S}}) \leq m^k n$ for TreeRes.

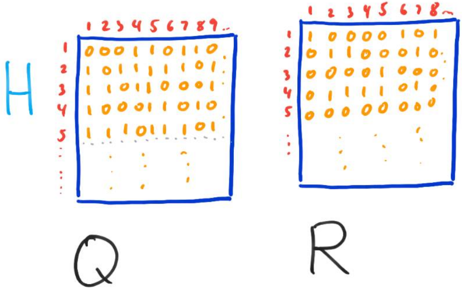
High-level idea: the universal property of A guarantees some column of Q will be a hitting set.

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

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

If $\gamma(\mathcal{S}) \leq k \leq \frac{\log m}{4}$, then $\tau_{\mathcal{S}}$ is unsatisfiable and $S(\tau_{\mathcal{S}}) \leq m^k n$ for TreeRes.

High-level idea: the universal property of A guarantees some column of Q will be a hitting set.

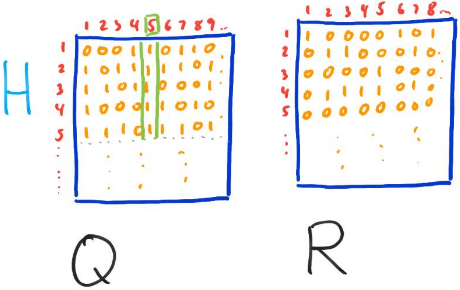


Upper bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

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

If $\gamma(\mathcal{S}) \leq k \leq \frac{\log m}{4}$, then $\tau_{\mathcal{S}}$ is unsatisfiable and $S(\tau_{\mathcal{S}}) \leq m^k n$ for TreeRes.

High-level idea: the universal property of A guarantees some column of Q will be a hitting set.

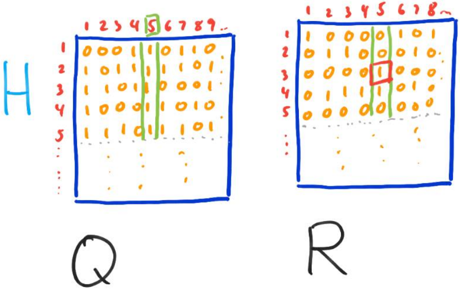


Upper bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

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

If $\gamma(\mathcal{S}) \leq k \leq \frac{\log m}{4}$, then $\tau_{\mathcal{S}}$ is unsatisfiable and $S(\tau_{\mathcal{S}}) \leq m^k n$ for TreeRes.

High-level idea: the universal property of A guarantees some column of Q will be a hitting set.

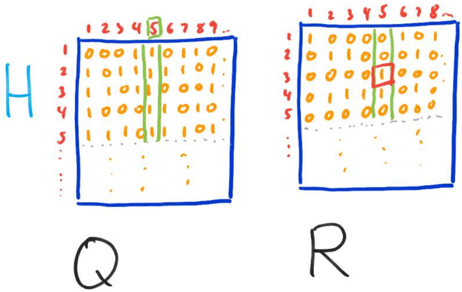


Upper bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

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

If $\gamma(\mathcal{S}) \leq k \leq \frac{\log m}{4}$, then $\tau_{\mathcal{S}}$ is unsatisfiable and $S(\tau_{\mathcal{S}}) \leq m^k n$ for TreeRes.

High-level idea: the universal property of A guarantees some column of Q will be a hitting set.



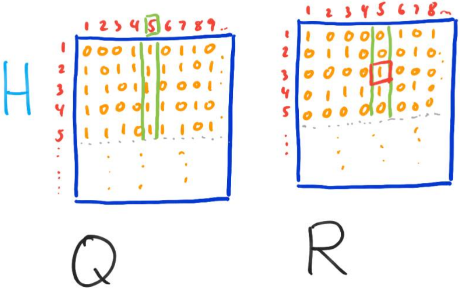
Size of the proof: $m^k n$

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

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

If $\gamma(\mathcal{S}) \leq k \leq \frac{\log m}{4}$, then $\tau_{\mathcal{S}}$ is unsatisfiable and $S(\tau_{\mathcal{S}}) \leq n^2$ for TreeRes.

High-level idea: the universal property of A guarantees some column of Q will be a hitting set.



Size of the proof: $m^k n = n^2$ for $m = n^{1/k}$

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

High-level idea 1: any proof π must query all rows in some hitting set

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

High-level idea 1: any proof π must query all rows in some hitting set

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

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

High-level idea 1: any proof π must query all rows in some hitting set

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

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

High-level idea 1: any proof π must query all rows in some hitting set

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

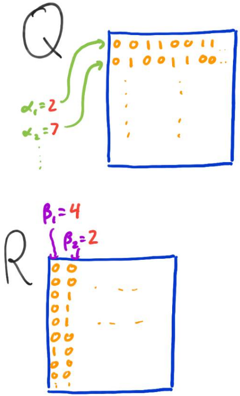
Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

High-level idea 1: any proof π must query all rows in some hitting set

- Res/TreeRes - prover-delayer game [Pudlák, Atserias-Lauria-Nordström]
- Nullsatz/PC - linear operator [Galesi-Lauria]
- Res(k) - switching lemma [Buss-Impagliazzo-Segerlend]
- TreeCP - lifting [upcoming work]

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

High-level idea 2: π knows nothing about a row or column without setting lots of variables

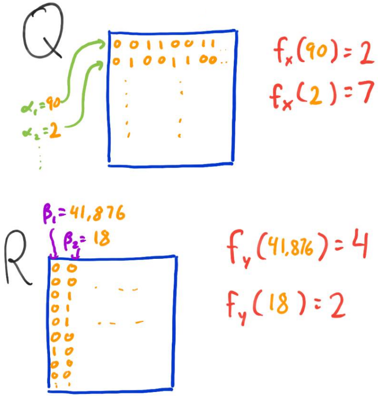


Lower bound on $S_{\mathcal{P}}(\tau_{\mathcal{S}})$

High-level idea 2: π knows nothing about a row or column without setting lots of variables

Error-correcting codes

- $x_i \in \{0,1\}^{6 \log m}$, $y_j \in \{0,1\}^{6 \log n}$
- $f_x : \{0,1\}^{6 \log m} \to [m]$, $f_y : \{0,1\}^{6 \log n} \to [n]$



Open problems

Better hard k in gap hitting set $\rightarrow$ better non-automatizability result

Open problems

Better hard k in gap hitting set $\rightarrow$ better non-automatizability result

Theorem (Chen-Lin)

Assuming ETH *the gap hitting set problem cannot be solved in time* $n^{o(k)}$ *for* $k = O(\log^{1/7-o(1)} \log n)$

Theorem (Main Technical Lemma)

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

Thank you!

~~ɔ:ˈtɒmətaɪzəˈbɪlɪti~~ *ɔ:tɒˈmætaɪzəˈbɪlɪti*