On the Virtue of Succinct Proofs: Amplifying Communication Complexity Hardness to Time-Space Trade-offs in Proof Complexity

Jakob Nordström

KTH Royal Institute of Technology
Stockholm, Sweden

Limits of Theorem Proving
Rome, Italy
September 25–27, 2012

Joint work with Trinh Huynh

The SAT Problem in Theory and Practice

- SAT NP-complete and so probably intractable in worst case
- But enormous progress on applied algorithms last 10-15 years
- Surprising fact 1: State-of-the-art SAT solvers can deal with real-world instances containing millions of variables
- Surprising fact 2: Best SAT solvers today still based on methods from early 1960s (i.e., DPLL and resolution)
- Algebraic and geometric methods more efficient in theory but **not** so far in practice

SAT Solving and Proof Complexity

SAT solving

- Constructive (almost deterministic) algorithms
- Key resources for solvers: time and memory
- Ideally minimize simultaneously

Proof complexity

- Study proofs, i.e., nondeterministic algorithms
- Complexity measures: proof size and proof space
- Lower bounds for optimal algorithms

SAT Solving and Proof Complexity

SAT solving

- Constructive (almost deterministic) algorithms
- Key resources for solvers: time and memory
- Ideally minimize simultaneously

Proof complexity

- Study proofs, i.e., nondeterministic algorithms
- Complexity measures: proof size and proof space
- Lower bounds for optimal algorithms

Hope to understand potential and limitation of SAT solvers by studying corresponding proof systems

Complexity measures also natural and interesting in their own right

This talk: Size-space trade-offs for algebraic and geometric systems

Outline

1. Proof Complexity
   - Preliminaries
   - Previous Work
   - Our Results

2. Tools and Techniques
   - Pebbling
   - Communication Complexity
   - Lifting
   - Critical Block Sensitivity

3. Open Problems

Some Terminology and Notation

- Literal a: variable x or its negation $\overline{x}$
- Clause $C = a_1 \lor \cdots \lor a_k$: disjunction of literals
  (Consider as sets, so no repetitions and order irrelevant)
- CNF formula $F = C_1 \land \cdots \land C_m$: conjunction of clauses
- k-CNF formula: all clauses of size $\leq k = \mathcal{O}(1)$
- Goal: Refute given CNF formula (i.e., prove it is unsatisfiable)
- Refer to clauses of CNF formula as axioms
  (as opposed to conclusions derived from these clauses)
- All formulas in this talk are k-CNFs
  (cleanest and most interesting case)

The Theoretical Model

- Proof system operates with lines of some syntactic form
- Proof/refutation is "presented on blackboard"
- Derivation steps:
  - Write down axiom clauses of CNF formula being refuted (as encoded by proof system)
  - Infer new lines by deductive rules of proof system
  - Erase lines not currently needed (to save space on blackboard)
- Refutation ends when contradiction is derived

The Theoretical Model

- Proof system operates with lines of some syntactic form
- Proof/refutation is “presented on blackboard”
- Derivation steps:
  - Write down axiom clauses of CNF formula being refuted (as encoded by proof system)
  - Infer new lines by deductive rules of proof system
  - Erase lines not currently needed (to save space on blackboard)
- Refutation ends when contradiction is derived

The Theoretical Model

- Proof system operates with lines of some syntactic form
- Proof/refutation is "presented on blackboard"
- Derivation steps:
  - Write down axiom clauses of CNF formula being refuted (as encoded by proof system)
  - Infer new lines by deductive rules of proof system
  - Erase lines not currently needed (to save space on blackboard)
- Refutation ends when contradiction is derived

$x \lor \overline{y} \lor z$

The Theoretical Model

- Proof system operates with lines of some syntactic form
- Proof/refutation is "presented on blackboard"
- Derivation steps:
  - Write down axiom clauses of CNF formula being refuted (as encoded by proof system)
  - Infer new lines by deductive rules of proof system
  - Erase lines not currently needed (to save space on blackboard)
- Refutation ends when contradiction is derived

$$x \lor \overline{y} \lor z$$
$$\overline{z} \lor \overline{u} \lor w$$

The Theoretical Model

- Proof system operates with lines of some syntactic form
- Proof/refutation is "presented on blackboard"
- Derivation steps:
  - Write down axiom clauses of CNF formula being refuted (as encoded by proof system)
  - Infer new lines by deductive rules of proof system
  - Erase lines not currently needed (to save space on blackboard)
- Refutation ends when contradiction is derived

$$x \lor \overline{y} \lor z$$
$$\overline{z} \lor \overline{u} \lor w$$
$$x \lor \overline{y} \lor \overline{u} \lor w$$

The Theoretical Model

- Proof system operates with lines of some syntactic form
- Proof/refutation is "presented on blackboard"
- Derivation steps:
  - Write down axiom clauses of CNF formula being refuted (as encoded by proof system)
  - Infer new lines by deductive rules of proof system
  - Erase lines not currently needed (to save space on blackboard)
- Refutation ends when contradiction is derived

$$x \lor \overline{y} \lor z$$
$$\overline{z} \lor \overline{u} \lor w$$
$$x \lor \overline{y} \lor \overline{u} \lor w$$

The Theoretical Model

- Proof system operates with lines of some syntactic form
- Proof/refutation is "presented on blackboard"
- Derivation steps:
  - Write down axiom clauses of CNF formula being refuted (as encoded by proof system)
  - Infer new lines by deductive rules of proof system
  - Erase lines not currently needed (to save space on blackboard)
- Refutation ends when contradiction is derived

$$\overline{z} \lor \overline{u} \lor w$$
$$x \lor \overline{y} \lor \overline{u} \lor w$$

The Theoretical Model

- Proof system operates with lines of some syntactic form
- Proof/refutation is “presented on blackboard”
- Derivation steps:
  - Write down axiom clauses of CNF formula being refuted (as encoded by proof system)
  - Infer new lines by deductive rules of proof system
  - Erase lines not currently needed (to save space on blackboard)
- Refutation ends when contradiction is derived

$\overline{z} \lor \overline{u} \lor w$

$x \lor \overline{y} \lor \overline{u} \lor w$

The Theoretical Model

- Proof system operates with lines of some syntactic form
- Proof/refutation is "presented on blackboard"
- Derivation steps:
  - Write down axiom clauses of CNF formula being refuted (as encoded by proof system)
  - Infer new lines by deductive rules of proof system
  - Erase lines not currently needed (to save space on blackboard)
- Refutation ends when contradiction is derived

$$x \lor \overline{y} \lor \overline{u} \lor w$$

Complexity Measures: Length, Size and Space

Length

\# derivation steps

Size

$\approx$ total # symbols in proof counted with repetitions

Space

$\approx$ max size of blackboard to carry out proof
(e.g., space 3 for this blackboard)

$$x \lor \overline{y} \lor z$$
$$\overline{z} \lor \overline{u} \lor w$$
$$x \lor \overline{y} \lor \overline{u} \lor w$$

Complexity Measures: Length, Size and Space

Length

$\#$ derivation steps

Size

$\approx$ total $\#$ symbols in proof counted with repetitions

Space

$\approx$ max size of blackboard to carry out proof
(e.g., space 3 for this blackboard)

$$x \lor \overline{y} \lor z$$
$$\overline{z} \lor \overline{u} \lor w$$
$$x \lor \overline{y} \lor \overline{u} \lor w$$

Note that:

1. These are somewhat informal definitions — see paper for (standard) details
2. Length and size can be very different — won't really distinguish between them too much in this talk

Resolution

Basis for the most successful SAT solvers to date
(DPLL method plus clause learning; a.k.a. CDCL)

Lines in refutation are disjunctive clauses

Resolution

Basis for the most successful SAT solvers to date
(DPLL method plus clause learning; a.k.a. CDCL)

Lines in refutation are disjunctive clauses

Resolution rule $\dfrac{C \lor x \qquad D \lor \overline{x}}{C \lor D}$

Resolution

Basis for the most successful SAT solvers to date
(DPLL method plus clause learning; a.k.a. CDCL)

Lines in refutation are disjunctive clauses

Resolution rule $\dfrac{C \lor x \qquad D \lor \overline{x}}{C \lor D}$

- Optimal (exponential) lower bounds on size
  [Urquhart '87; Chvátal & Szemerédi '88]
- Optimal (linear) lower bounds on clause space
  [Torán '99; Alekhnovich, Ben-Sasson, Razborov & Wigderson '00]
- Strong size-space trade-offs
  [Ben-Sasson & N. '11; Beame, Beck & Impagliazzo '12]

Polynomial Calculus (or Actually PCR [ABRW '00])

Clauses interpreted as polynomial equations over finite field

E.g., $x \lor y \lor \overline{z}$ translated to $x'y'z = 0$

Show no common root by deriving $1 = 0$

Polynomial Calculus (or Actually PCR [ABRW '00])

Clauses interpreted as polynomial equations over finite field
E.g., $x \vee y \vee \overline{z}$ translated to $x'y'z = 0$
Show no common root by deriving $1 = 0$

Boolean axioms $\dfrac{}{x^2 - x = 0}$ *Negation* $\dfrac{}{x + x' = 1}$

Linear combination $\dfrac{p = 0 \quad q = 0}{\alpha p + \beta q = 0}$ *Multiplication* $\dfrac{p = 0}{xp = 0}$

Polynomial Calculus (or Actually PCR [ABRW '00])

Clauses interpreted as polynomial equations over finite field
E.g., $x \vee y \vee \overline{z}$ translated to $x'y'z = 0$
Show no common root by deriving $1 = 0$

Boolean axioms $\dfrac{}{x^2 - x = 0}$

Negation $\dfrac{}{x + x' = 1}$

Linear combination $\dfrac{p = 0 \qquad q = 0}{\alpha p + \beta q = 0}$

Multiplication $\dfrac{p = 0}{xp = 0}$

- Optimal (exponential) lower bounds on size [Alekhnovich-Razborov '01] and others
- Only recently lower bounds on monomial space for k-CNFs [Filmus, Lauria, N., Ron-Zewi & Thapen '12] building on [ABRW '00]
  Very recent optimal bounds in [Bonacina & Galesi '13]
- No size-space trade-offs

Cutting Planes

Clauses interpreted as linear inequalities
E.g., $x \lor y \lor \overline{z}$ translated to $x + y + (1 - z) \geq 1$
Show inconsistent by deriving $0 \geq 1$

Cutting Planes

Clauses interpreted as linear inequalities
E.g., $x \vee y \vee \overline{z}$ translated to $x + y + (1 - z) \geq 1$
Show inconsistent by deriving $0 \geq 1$

Variable axioms $\dfrac{}{0 \leq x \leq 1}$

Multiplication $\dfrac{\sum a_i x_i \geq A}{\sum c a_i x_i \geq cA}$

Addition $\dfrac{\sum a_i x_i \geq A \quad \sum b_i x_i \geq B}{\sum (a_i + b_i) x_i \geq A + B}$

Division $\dfrac{\sum c a_i x_i \geq A}{\sum a_i x_i \geq \lceil A/c \rceil}$

Cutting Planes

Clauses interpreted as linear inequalities
E.g., $x \lor y \lor \overline{z}$ translated to $x + y + (1 - z) \geq 1$
Show inconsistent by deriving $0 \geq 1$

Variable axioms $\dfrac{}{0 \leq x \leq 1}$

Multiplication $\dfrac{\sum a_i x_i \geq A}{\sum c a_i x_i \geq cA}$

Addition $\dfrac{\sum a_i x_i \geq A \quad \sum b_i x_i \geq B}{\sum (a_i + b_i) x_i \geq A + B}$

Division $\dfrac{\sum c a_i x_i \geq A}{\sum a_i x_i \geq \lceil A/c \rceil}$

- Only one (exponential) lower bounds on size [Pudlák '97]
- No lower bounds on line space
- No size-space trade-offs

Trade-offs for Polynomial Calculus and Cutting Planes

We make some progress on understanding space and size-space trade-offs in polynomial calculus and cutting planes

Theorem (Informal)

There are k-CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ such that

- *resolution can refute F_n in length $\mathcal{O}(n)$ (and hence so can polynomial calculus and cutting planes)*
- *any polynomial calculus or cutting planes refutation of F_n in length L and space s must have*

$$s \log L \gtrsim \sqrt[4]{n}$$

Trade-offs for Polynomial Calculus and Cutting Planes

We make some progress on understanding space and size-space trade-offs in polynomial calculus and cutting planes

Theorem (Informal)

There are k-CNF formulas $\{F_n\}_{n=1}^{\infty}$ of size $\Theta(n)$ such that

- *resolution can refute F_n in length $\mathcal{O}(n)$ (and hence so can polynomial calculus and cutting planes)*
- *any polynomial calculus or cutting planes refutation of F_n in length L and space s must have*

$$s \log L \gtrsim \sqrt[4]{n}$$

Nice bonus: lower bounds hold for semantic versions of proof systems where anything implied by blackboard can be inferred in just one step

Proof Ingredients

- Pebbling
- Communication complexity
- Lifting
- Critical block sensitivity

How to Get a Handle on Time-Space Relations?

Questions about time-space trade-offs fundamental in theoretical computer science

How to Get a Handle on Time-Space Relations?

Questions about time-space trade-offs fundamental in theoretical computer science

In particular, well-studied (and well-understood) for pebble games modelling calculations described by DAGs ([Cook & Sethi '76] and many others)

- Time needed for calculation: # pebbling moves
- Space needed for calculation: max # pebbles required

How to Get a Handle on Time-Space Relations?

Questions about time-space trade-offs fundamental in theoretical computer science

In particular, well-studied (and well-understood) for pebble games modelling calculations described by DAGs ([Cook & Sethi '76] and many others)

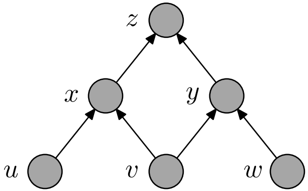
- Time needed for calculation: # pebbling moves
- Space needed for calculation: max # pebbles required

Some quick graph terminology

- DAGs consist of vertices with directed edges between them
- vertices with no incoming edges: sources
- vertices with no outgoing edges: sinks

The Black-White Pebble Game

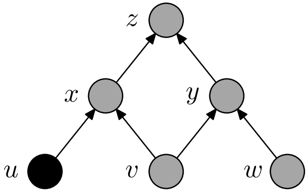
Goal: get single black pebble on sink vertex z of G



# moves	0
Current # pebbles	0
Max # pebbles so far	0

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

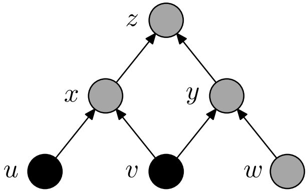


# moves	1
Current # pebbles	1
Max # pebbles so far	1

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

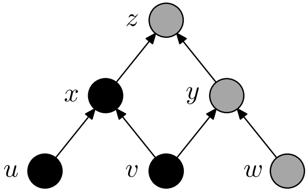


# moves	2
Current # pebbles	2
Max # pebbles so far	2

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

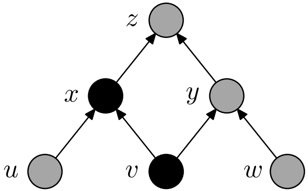


# moves	3
Current # pebbles	3
Max # pebbles so far	3

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

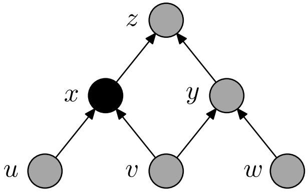


# moves	4
Current # pebbles	2
Max # pebbles so far	3

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them
2. Can always remove black pebble from vertex

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

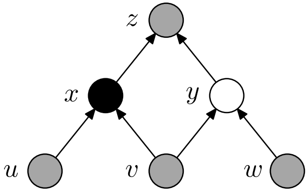


# moves	5
Current # pebbles	1
Max # pebbles so far	3

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them
2. Can always remove black pebble from vertex

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

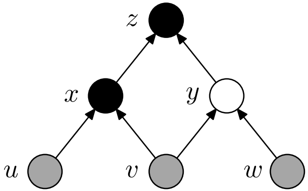


# moves	6
Current # pebbles	2
Max # pebbles so far	3

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them
2. Can always remove black pebble from vertex
3. Can always place white pebble on (empty) vertex

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

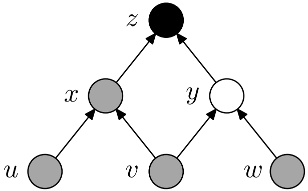


# moves	7
Current # pebbles	3
Max # pebbles so far	3

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them
2. Can always remove black pebble from vertex
3. Can always place white pebble on (empty) vertex

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

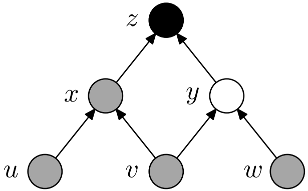


# moves	8
Current # pebbles	2
Max # pebbles so far	3

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them
2. Can always remove black pebble from vertex
3. Can always place white pebble on (empty) vertex

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

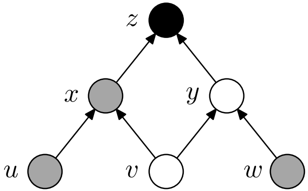


# moves	8
Current # pebbles	2
Max # pebbles so far	3

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them
2. Can always remove black pebble from vertex
3. Can always place white pebble on (empty) vertex
4. Can remove white pebble if all predecessors have pebbles

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

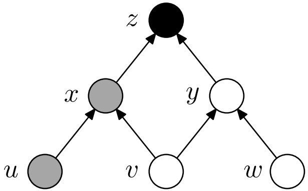


# moves	9
Current # pebbles	3
Max # pebbles so far	3

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them
2. Can always remove black pebble from vertex
3. Can always place white pebble on (empty) vertex
4. Can remove white pebble if all predecessors have pebbles

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

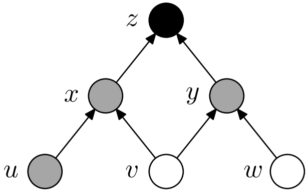


# moves	10
Current # pebbles	4
Max # pebbles so far	4

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them
2. Can always remove black pebble from vertex
3. Can always place white pebble on (empty) vertex
4. Can remove white pebble if all predecessors have pebbles

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

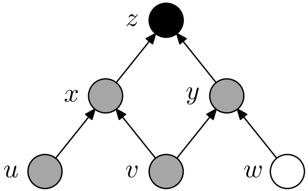


# moves	11
Current # pebbles	3
Max # pebbles so far	4

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them
2. Can always remove black pebble from vertex
3. Can always place white pebble on (empty) vertex
4. Can remove white pebble if all predecessors have pebbles

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

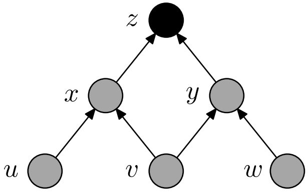


# moves	12
Current # pebbles	2
Max # pebbles so far	4

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them
2. Can always remove black pebble from vertex
3. Can always place white pebble on (empty) vertex
4. Can remove white pebble if all predecessors have pebbles

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G



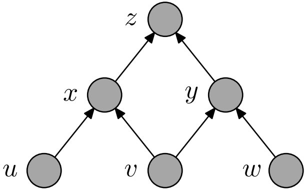
# moves	13
Current # pebbles	1
Max # pebbles so far	4

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them
2. Can always remove black pebble from vertex
3. Can always place white pebble on (empty) vertex
4. Can remove white pebble if all predecessors have pebbles

Pebbling Contradiction

CNF formulas encoding pebble game played on DAG G

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$

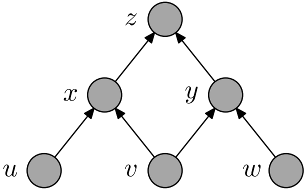


- sources are true
- truth propagates upwards
- but sink is false

Pebbling Contradiction

CNF formulas encoding pebble game played on DAG G

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

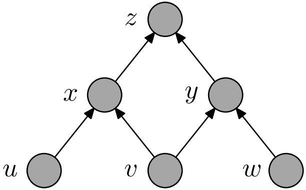


- sources are true
- truth propagates upwards
- but sink is false

Pebbling Contradiction

CNF formulas encoding pebble game played on DAG G

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

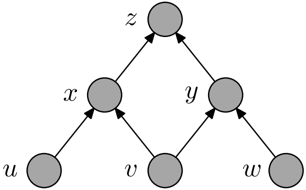


- sources are true
- truth propagates upwards
- but sink is false

Pebbling Contradiction

CNF formulas encoding pebble game played on DAG G

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

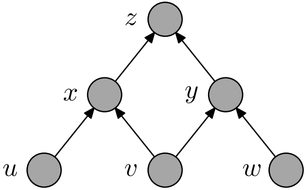


- sources are true
- truth propagates upwards
- but sink is false

Pebbling Contradiction

CNF formulas encoding pebble game played on DAG G

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$



- sources are true
- truth propagates upwards
- but sink is false

Appeared in various contexts in [Bonet et al. '98, Raz & McKenzie '99, Ben-Sasson & Wigderson '99] and other papers

Used to study size and space in resolution in [N. '06, N. & Håstad '08, Ben-Sasson & N. '08, '11]

Two-Player Randomized Communication Complexity

- Alice has private input x and private source of randomness
- Bob has private input y and private source of randomness
- Both have unbounded computational powers
- Want to compute $f(x, y)$ by sending messages back and forth
- Output correct for any x and y except with error probability ε
- Communication cost: max # bits communicated on any x and y

Falsified Clause Search Problem

Fix:

- unsatisfiable CNF formula F
- (devious) partition of $Vars(F)$ between Alice and Bob

Falsified clause search problem $Search(F)$

Input: Assignment α to $Vars(F)$ split between Alice and Bob

Output: Clause $C \in F$ such that $\alpha(C) = 0$

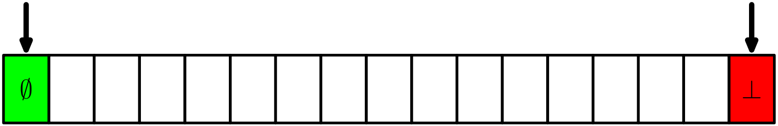
Actually, computing not function but relation — more about that later

Succinct Refutations Yield Efficient Protocols

Evaluate blackboard configurations of a refutation of F under α

Succinct Refutations Yield Efficient Protocols

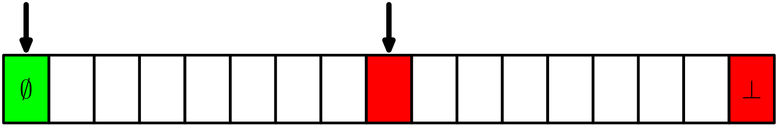
Evaluate blackboard configurations of a refutation of F under α



Use binary search to find transition from true to false blackboard

Succinct Refutations Yield Efficient Protocols

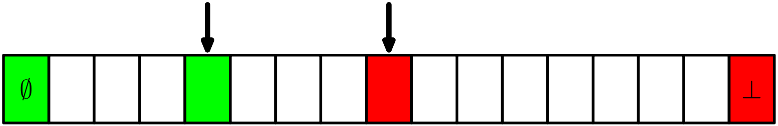
Evaluate blackboard configurations of a refutation of F under α



Use binary search to find transition from true to false blackboard

Succinct Refutations Yield Efficient Protocols

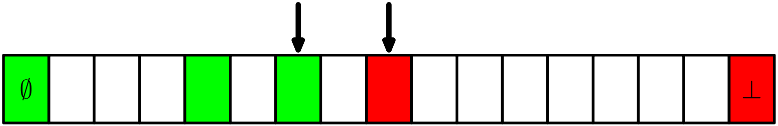
Evaluate blackboard configurations of a refutation of F under α



Use binary search to find transition from true to false blackboard

Succinct Refutations Yield Efficient Protocols

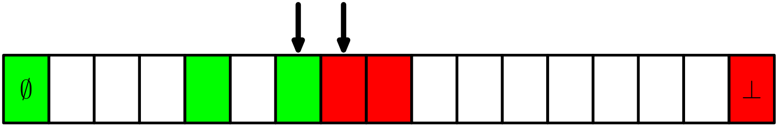
Evaluate blackboard configurations of a refutation of F under α



Use binary search to find transition from true to false blackboard

Succinct Refutations Yield Efficient Protocols

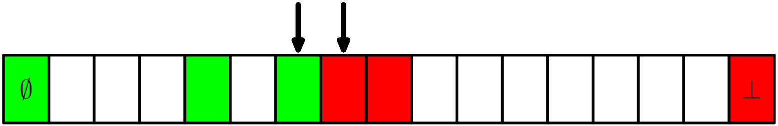
Evaluate blackboard configurations of a refutation of F under α



Use binary search to find transition from true to false blackboard

Succinct Refutations Yield Efficient Protocols

Evaluate blackboard configurations of a refutation of F under α

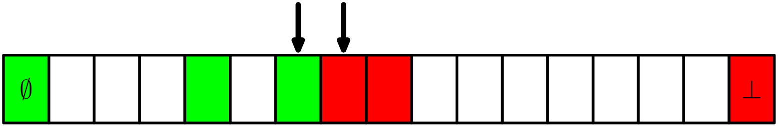


Use binary search to find transition from true to false blackboard

Must happen when $C \in F$ written down — answer to $Search(F)$

Succinct Refutations Yield Efficient Protocols

Evaluate blackboard configurations of a refutation of F under α



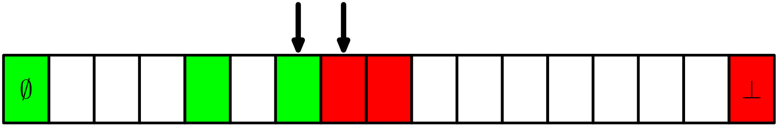
Use binary search to find transition from true to false blackboard

Must happen when $C \in F$ written down — answer to $Search(F)$

Refutation length $L \Rightarrow$ evaluate $\log L$ blackboards

Succinct Refutations Yield Efficient Protocols

Evaluate blackboard configurations of a refutation of F under α



Use binary search to find transition from true to false blackboard

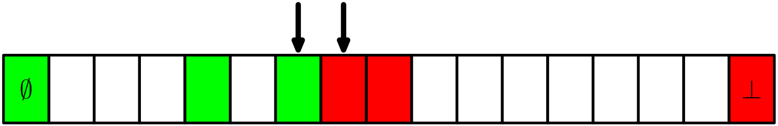
Must happen when $C \in F$ written down — answer to $Search(F)$

Refutation length $L \Rightarrow$ evaluate $\log L$ blackboards

Refutation space $s \Rightarrow$ max $\approx s$ bits of communication per blackboard

Succinct Refutations Yield Efficient Protocols

Evaluate blackboard configurations of a refutation of F under α



Use binary search to find transition from true to false blackboard

Must happen when $C \in F$ written down — answer to $Search(F)$

Refutation length $L \Rightarrow$ evaluate $\log L$ blackboards

Refutation space $s \Rightarrow$ max $\approx s$ bits of communication per blackboard

(E.g. for polynomial calculus Alice and Bob simply evaluate their part of each monomial and exchange values — cutting planes bit more involved but can be done)

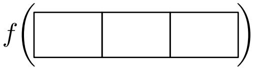
Lifting of Functions

Construct hard communication problems by "hardness amplification" using lifting

Lifting of Functions

Construct hard communication problems by “hardness amplification” using lifting

Start with function
$f : \{0,1\}^m \mapsto \{0,1\}$

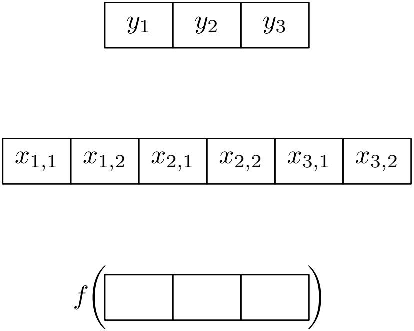


Lifting of Functions

Construct hard communication problems by "hardness amplification" using lifting

Start with function
$f : \{0,1\}^m \mapsto \{0,1\}$

Construct new function on inputs
$x \in \{0,1\}^{\ell m}$ and $y \in [\ell]^m$



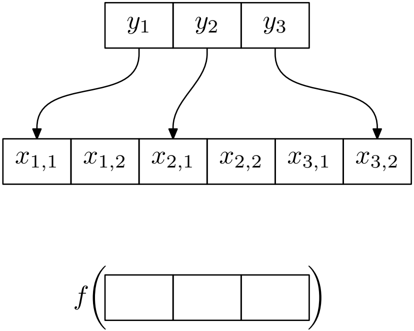
Lifting of Functions

Construct hard communication problems by "hardness amplification" using lifting

Start with function
$f : \{0,1\}^m \mapsto \{0,1\}$

Construct new function on inputs
$x \in \{0,1\}^{\ell m}$ and $y \in [\ell]^m$

Alice's y-variables determine. . .



Lifting of Functions

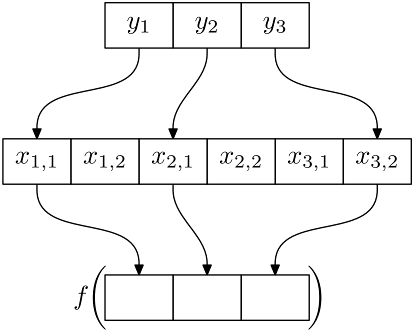
Construct hard communication problems by "hardness amplification" using lifting

Start with function
$f : \{0,1\}^m \mapsto \{0,1\}$

Construct new function on inputs
$x \in \{0,1\}^{\ell m}$ and $y \in [\ell]^m$

Alice's y-variables determine...

...which of Bob's x-bits to feed to f



Lifting of Functions

Construct hard communication problems by "hardness amplification" using lifting

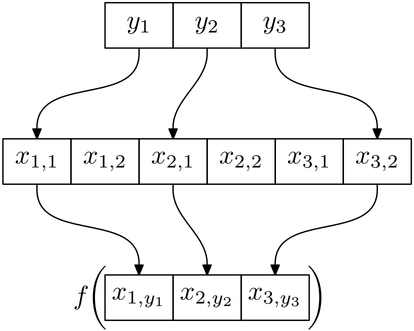
Start with function
$f : \{0,1\}^m \mapsto \{0,1\}$

Construct new function on inputs
$x \in \{0,1\}^{\ell m}$ and $y \in [\ell]^m$

Alice's y-variables determine. . .

. . . which of Bob's x-bits to feed to f

Length-ℓ lifting of f defined as
$Lift_\ell(f)(x,y) := f(x_{1,y_1}, \ldots, x_{m,y_m})$



Lifting of Functions

Construct hard communication problems by "hardness amplification" using lifting

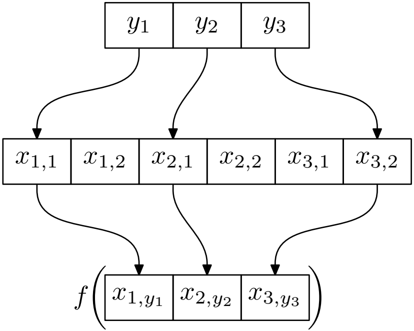
Start with function $f : \{0,1\}^m \mapsto \{0,1\}$

Construct new function on inputs $x \in \{0,1\}^{\ell m}$ and $y \in [\ell]^m$

Alice's y-variables determine...

...which of Bob's x-bits to feed to f

Length-ℓ lifting of f defined as $Lift_\ell(f)(x,y) := f(x_{1,y_1}, \ldots, x_{m,y_m})$



Idea borrowed from [Beame, Huynh & Pitassi '10]

Critical Block Sensitivity of Search Problems

- Block sensitivity of f on α: $\#$ disjoint blocks of α that flip f if flipped
- Problem: falsified clause search problem defines relation, not function
- Study block sensitivity of search problems
- In addition restrict to critical inputs (where relation is “function-like” in that there is only one right answer)
- Prove randomized communication complexity lower bounds in terms of critical block sensitivity of search problems
- Proof uses information-theoretic approach inspired by [Bar-Yossef, Jayram, Kumar & Sivakumar ’04]

Communication Complexity Results

We prove two technical lemmas:

Lemma 1

If critical block sensitivity of search problem S is large, then communication complexity of lifted search problem $Lift(S)$ is large

Communication Complexity Results

We prove two technical lemmas:

Lemma 1

If critical block sensitivity of search problem S is large, then communication complexity of lifted search problem $\mathit{Lift}(S)$ is large

Lemma 2

Search problems for pebbling formulas constructed from specific family of pyramid graphs have large critical block sensitivity

Putting the Pieces Together

- Encode lifting of search problem for CNF as new formula $\mathit{Lift}(F)$ (as in [Beame, Huynh & Pitassi '10])

Putting the Pieces Together

- Encode lifting of search problem for CNF as new formula $Lift(F)$ (as in [Beame, Huynh & Pitassi '10])
- Efficient refutation of $Lift(F)$
  $\Rightarrow$ efficient communication protocol for $Search(Lift(F))$

Putting the Pieces Together

- Encode lifting of search problem for CNF as new formula $\mathit{Lift}(F)$ (as in [Beame, Huynh & Pitassi '10])
- Efficient refutation of $\mathit{Lift}(F)$
  $\Rightarrow$ efficient communication protocol for $\mathit{Search}(\mathit{Lift}(F))$
- Protocol for $\mathit{Search}(\mathit{Lift}(F))$
  $\Rightarrow$ use to solve $\mathit{Lift}(\mathit{Search}(F))$ — easy

Putting the Pieces Together

- Encode lifting of search problem for CNF as new formula $\mathit{Lift}(F)$ (as in [Beame, Huynh & Pitassi '10])
- Efficient refutation of $\mathit{Lift}(F)$
  $\Rightarrow$ efficient communication protocol for $\mathit{Search}(\mathit{Lift}(F))$
- Protocol for $\mathit{Search}(\mathit{Lift}(F))$
  $\Rightarrow$ use to solve $\mathit{Lift}(\mathit{Search}(F))$ — easy
- But communication complexity of lifted search problem lower-bounded by critical block sensitivity (Lemma 1)

Putting the Pieces Together

- Encode lifting of search problem for CNF as new formula $Lift(F)$ (as in [Beame, Huynh & Pitassi '10])
- Efficient refutation of $Lift(F)$
  $\Rightarrow$ efficient communication protocol for $Search(Lift(F))$
- Protocol for $Search(Lift(F))$
  $\Rightarrow$ use to solve $Lift(Search(F))$ — easy
- But communication complexity of lifted search problem lower-bounded by critical block sensitivity (Lemma 1)
- Plug in lower bound for pyramid pebbling formulas (Lemma 2)
  $\Rightarrow$ trade-off for lifted pebbling formulas

Block Sensitivity in More Detail

Block sensitivity of f at α:

$$f\left(\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|}\hline 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ \hline\end{array}\right) = 0$$

Block Sensitivity in More Detail

Block sensitivity of f at α:

$$f\left(\boxed{0 \mid 0 \mid 0 \mid 1 \mid 1 \mid 0 \mid 1 \mid 0 \mid 0 \mid 0 \mid 1 \mid 0}\right) = 1$$

Block Sensitivity in More Detail

Block sensitivity of f at α:

$$f\left(\boxed{0\ |\ \mathbf{1\ |\ 1\ |\ 0}\ |\ 1\ |\ 0\ |\ 1\ |\ 0\ |\ 0\ |\ 0\ |\ 1\ |\ 0}\right) = 0$$

Block Sensitivity in More Detail

Block sensitivity of f at α:

$$f\left(\boxed{0}\,\boxed{1\;1\;0}\,\boxed{0\;1\;0}\,\boxed{0}\,\boxed{0}\,\boxed{0}\,\boxed{1}\,\boxed{0}\right) = 1$$

Block Sensitivity in More Detail

Block sensitivity of f at α:

$$f\left(\boxed{0\,|\,1\,|\,1\,|\,0\,|\,1\,|\,0\,|\,1\,|\,0\,|\,0\,|\,0\,|\,1\,|\,0}\right) = 0$$

Block Sensitivity in More Detail

Block sensitivity of f at α:

$$f\left(\boxed{0}\,\boxed{1\;1\;0}\,\boxed{1\;0\;1}\,0\,\boxed{\color{red}1}\,0\;1\;0\right) = \color{red}{1}$$

Block Sensitivity in More Detail

Block sensitivity of f at α:

$$f\left(\boxed{0 \mid 1 \mid 1 \mid 0 \mid 1 \mid 0 \mid 1 \mid 0 \mid 0 \mid 0 \mid 1 \mid 0}\right) = 0$$

Block Sensitivity in More Detail

Block sensitivity of f at α:

$$f\left(\boxed{0\;1\;1\;0\;1\;0\;1\;0\;0\;0\;1\;0}\right) = 0$$

$bs(f, \alpha) = 3$ in this example

Block Sensitivity in More Detail

Block sensitivity of f at α:

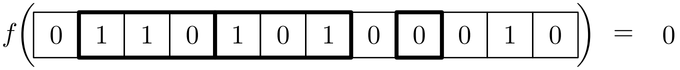
$$f\left(\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|}\hline 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ \hline\end{array}\right) = 0$$

$bs(f, \alpha) = 3$ in this example

$bs(f, \mathcal{A})$: largest block sensitivity for any α in subset $\mathcal{A}$ of inputs

Block Sensitivity in More Detail

Block sensitivity of f at α:



$bs(f, \alpha) = 3$ in this example

$bs(f, \mathcal{A})$: largest block sensitivity for any α in subset $\mathcal{A}$ of inputs

f solves search problem $S \subseteq \{0,1\}^m \times Q$ if it holds that $(\alpha, f(\alpha)) \in S$

Block Sensitivity in More Detail

Block sensitivity of f at α:

$$f\big(\boxed{0}\,\boxed{1\,1\,0}\,\boxed{1\,0\,1}\,\boxed{0}\,\boxed{0}\,\boxed{0}\,\boxed{1}\,\boxed{0}\big) = 0$$

$bs(f, \alpha) = 3$ in this example

$bs(f, \mathcal{A})$: largest block sensitivity for any α in subset $\mathcal{A}$ of inputs

f solves search problem $S \subseteq \{0, 1\}^m \times Q$ if it holds that $(\alpha, f(\alpha)) \in S$

$bs_{crit}(S)$: block sensitivity over critical assignments $\mathcal{A}$ of best f solving S

Lifting and Critical Block Sensitivity

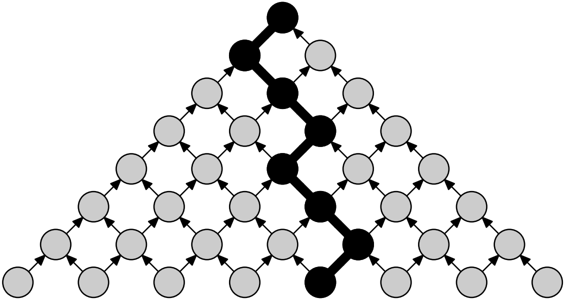
Lemma 2 (more formal version)

Suppose $S \subseteq \{0,1\}^m \times Q$ is a search problem and $\ell \geq 3$. Then any consistent randomized protocol solving $\mathit{Lift}_\ell(S)$, where Alice receives the selector y-variables and Bob receives the main x-variables, requires $\Omega(bs_{crit}(S))$ bits of communication.

Proof is by

- information theory tools
- direct sum theorem à la [BJKS04]

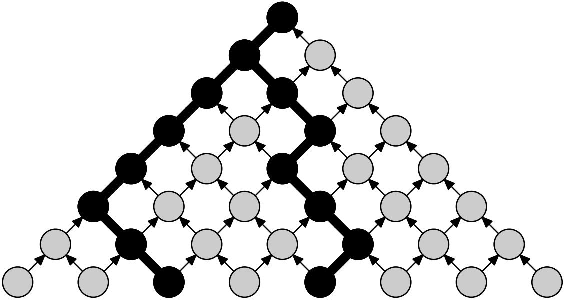
Critical Assignments for Pyramid Pebbling Formulas



Focus on critical assignment setting:

- vertices on one source-to-sink path P false
- all other vertices true (so $\mathrm{source}(P)$ only correct answer)

Critical Assignments for Pyramid Pebbling Formulas



Focus on critical assignment setting:

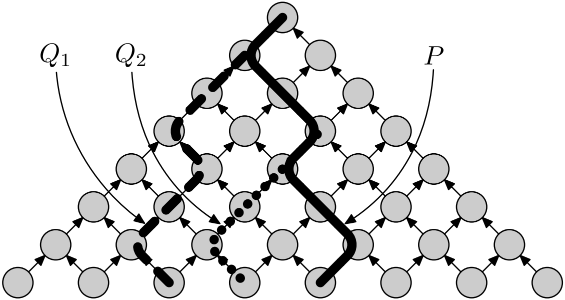
- vertices on one source-to-sink path P false
- all other vertices true (so $\mathrm{source}(P)$ only correct answer)

Bicritical assignments falsify two different paths
$\Rightarrow$ two possible correct answers

Path Graph

Build graph G such that

- vertices = source-to-sink paths P
- edge (P, Q) only if P and Q merge and stay together
- in addition, if (P, Q_1) and (P, Q_2) edges, then $Q_1 \cap Q_2 \subseteq P$
- G is undirected — (P, Q) edge only if (Q, P) edge



Dense Path Graph $\Rightarrow$ High Critical Block Sensitivity

Lemma 3

If $\exists$ path graph G with average degree d, then falsified clause search problem for pebbling formula has critical block sensitivity $\geq d/2$

Dense Path Graph ⇒ High Critical Block Sensitivity

Lemma 3

If $\exists$ path graph G with average degree d, then falsified clause search problem for pebbling formula has critical block sensitivity $\geq d/2$

Proof:

- Orient G based on function f solving search problem

Dense Path Graph $\Rightarrow$ High Critical Block Sensitivity

Lemma 3

If $\exists$ path graph G with average degree d, then falsified clause search problem for pebbling formula has critical block sensitivity $\geq d/2$

Proof:

- Orient G based on function f solving search problem
- If f answers $\mathrm{source}(Q)$ for bicritical (P, Q), direct edge $P \rightarrow Q$

Dense Path Graph $\Rightarrow$ High Critical Block Sensitivity

Lemma 3

If $\exists$ path graph G with average degree d, then falsified clause search problem for pebbling formula has critical block sensitivity $\geq d/2$

Proof:

- Orient G based on function f solving search problem
- If f answers $\mathrm{source}(Q)$ for bicritical (P, Q), direct edge $P \rightarrow Q$
- Some P must have outdegree $\geq d/2$

Dense Path Graph $\Rightarrow$ High Critical Block Sensitivity

Lemma 3

If $\exists$ path graph G with average degree d, then falsified clause search problem for pebbling formula has critical block sensitivity $\geq d/2$

Proof:

- Orient G based on function f solving search problem
- If f answers $\mathrm{source}(Q)$ for bicritical (P, Q), direct edge $P \rightarrow Q$
- Some P must have outdegree $\geq d/2$
- When P flipped to bicritical (P, Q_i) for $P \rightarrow Q_i$, then f changes

Dense Path Graph $\Rightarrow$ High Critical Block Sensitivity

Lemma 3

If $\exists$ path graph G with average degree d, then falsified clause search problem for pebbling formula has critical block sensitivity $\geq d/2$

Proof:

- Orient G based on function f solving search problem
- If f answers $\mathrm{source}(Q)$ for bicritical (P, Q), direct edge $P \rightarrow Q$
- Some P must have outdegree $\geq d/2$
- When P flipped to bicritical (P, Q_i) for $P \rightarrow Q_i$, then f changes
- Hence critical block sensitivity $\geq d/2$, Q.E.D.

Dense Path Graph $\Rightarrow$ High Critical Block Sensitivity

Lemma 3

If $\exists$ path graph G with average degree d, then falsified clause search problem for pebbling formula has critical block sensitivity $\geq d/2$

Proof:

- Orient G based on function f solving search problem
- If f answers $\mathrm{source}(Q)$ for bicritical (P, Q), direct edge $P \to Q$
- Some P must have outdegree $\geq d/2$
- When P flipped to bicritical (P, Q_i) for $P \to Q_i$, then f changes
- Hence critical block sensitivity $\geq d/2$, Q.E.D.

Lemma 4

For pyramid on n vertices, can get average degree $\Omega\left(\sqrt[4]{n}\right)$

More General Trade-offs?

Our proofs only work for formulas generated from pyramid graphs

For resolution, correspondence between pebbling and size-space trade-offs holds for arbitrary graphs

More General Trade-offs?

Our proofs only work for formulas generated from pyramid graphs

For resolution, correspondence between pebbling and size-space trade-offs holds for arbitrary graphs

Open Problem

Can our trade-offs be extended to pebbling formulas over any graphs?

More General Trade-offs?

Our proofs only work for formulas generated from pyramid graphs

For resolution, correspondence between pebbling and size-space trade-offs holds for arbitrary graphs

Open Problem

Can our trade-offs be extended to pebbling formulas over any graphs?

Recently achieved for polynomial calculus in [Beck, N. & Tang '13]
Uses different techniques; in particular random restrictions
$\Rightarrow$ not tight results as for resolution, so room for further improvements

Still open for cutting planes (random restrictions don't work)

Unconditional Space Lower Bounds?

Open Problem

Can log *length factor be removed from results to yield unconditional space lower bounds?*

Unconditional Space Lower Bounds?

Open Problem

Can log *length factor be removed from results to yield unconditional space lower bounds?*

Again answer known to be "yes" for resolution

But [Beck, N. & Tang '12] still has log factor for polynomial calculus

Underlying question: For how wide a family of proof systems do pebbling properties of graphs carry over to CNF size-space trade-offs?

Take-Home Message

- Modern SAT solvers enormously successful in practice — key issue is to minimize time and memory consumption
- Modelled by proof size and space in proof complexity
- We show trade-offs indicating that simultaneous optimization impossible for well-known algebraic and geometric proof systems
- Future theoretical work: Understand size and space in these proof systems better
- Future practical work: Build efficient algebraic or geometric SAT solvers!

Take-Home Message

- Modern SAT solvers enormously successful in practice — key issue is to minimize time and memory consumption
- Modelled by proof size and space in proof complexity
- We show trade-offs indicating that simultaneous optimization impossible for well-known algebraic and geometric proof systems
- Future theoretical work: Understand size and space in these proof systems better
- Future practical work: Build efficient algebraic or geometric SAT solvers!

Thank you for your attention!