A Generalized Method for Proving Polynomial Calculus Degree Lower Bounds

Jakob Nordström

KTH Royal Institute of Technology
Stockholm, Sweden

Semidefinite and Matrix Methods for Optimization and Communication
Institute for Mathematical Sciences
National University of Singapore
February 4, 2016

Joint work with Mladen Mikša

The Satisfiability Problem (SAT)

$$(x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z})$$

The Satisfiability Problem (SAT)

$$(x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z})$$

- Variables should be set to true or false

The Satisfiability Problem (SAT)

$$(x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z})$$

- Variables should be set to true or false
- Constraint $(x \lor \overline{y} \lor z)$: means x or z should be true or y false

The Satisfiability Problem (SAT)

$$(x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z})$$

- Variables should be set to true or false
- Constraint $(x \lor \overline{y} \lor z)$: means x or z should be true or y false
- $\land$ means all constraints should hold simultaneously

The Satisfiability Problem (SAT)

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

- Variables should be set to true or false
- Constraint $(x \vee \overline{y} \vee z)$: means x or z should be true or y false
- $\wedge$ means all constraints should hold simultaneously

Is there a truth value assignment satisfying all these conditions?
Or is it always the case that some constraint must fail to hold?

Proof Complexity

Satisfiable formulas have short, efficiently verifiable certificates (satisfying assignments)

Proof Complexity

Satisfiable formulas have short, efficiently verifiable certificates (satisfying assignments)

What about unsatisfiable formulas?

Proof Complexity

Satisfiable formulas have short, efficiently verifiable certificates (satisfying assignments)

What about unsatisfiable formulas?

Proof system

Formal specification of method for reasoning about formulas
Given formula $\mathcal{F}$, can produce certificate π of unsatisfiability
Proof π should be polynomial-time verifiable (in size of π, not $\mathcal{F}$)

Proof Complexity

Satisfiable formulas have short, efficiently verifiable certificates (satisfying assignments)

What about unsatisfiable formulas?

Proof system

Formal specification of method for reasoning about formulas
Given formula $\mathcal{F}$, can produce certificate π of unsatisfiability
Proof π should be polynomial-time verifiable (in size of π, not $\mathcal{F}$)

Proof complexity

Study of upper and lower bounds for concrete proof systems

Motivations for Proof Complexity

Program for showing P $\neq$ NP

Original motivation in [Cook & Reckhow '79]

Superpolynomial lower bounds for all proof systems $\Rightarrow$ NP $\neq$ co-NP

Still very distant goal...

Motivations for Proof Complexity

Program for showing $\mathsf{P} \neq \mathsf{NP}$

Original motivation in [Cook & Reckhow '79]

Superpolynomial lower bounds for all proof systems $\Rightarrow$ NP $\neq$ co-NP

Still very distant goal...

Quantify power of mathematical reasoning

Study efficient proofs of different mathematical principles

Determine how strong proof systems are needed

Measures "mathematical depth" of corresponding principle

Motivations for Proof Complexity

Program for showing $\mathsf{P} \neq \mathsf{NP}$

Original motivation in [Cook & Reckhow '79]

Superpolynomial lower bounds for all proof systems $\Rightarrow$ NP $\neq$ co-NP

Still very distant goal...

Quantify power of mathematical reasoning

Study efficient proofs of different mathematical principles

Determine how strong proof systems are needed

Measures "mathematical depth" of corresponding principle

Connections to SAT solving and combinatorial optimization

Can formalize and study proof systems behind state-of-the-art SAT solvers

Sheds light on potential and limitations of such solvers

Also extends to combinatorial optimization (e.g., LP and SDP hierarchies)

Outline of This Presentation

1. Overview of some proof complexity basics
2. Discuss two proof systems
   - Resolution ($\Leftrightarrow$ state-of-the-art conflict-driven clause learning solvers)
   - Polynomial calculus ($\Leftrightarrow$ algebraic Gröbner basis computations)
3. Present framework for proving polynomial calculus lower bounds
   - Based on degree lower bounds via expansion
   - Expressed in terms of combinatorial game played on formula
   - Unifies previous lower bounds and yields some new ones

Some Notation and Terminology

- 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: CNF formula with clauses of size $\leq k$
  (where k is some constant)
- $N =$ size of formula (# literals, which is $\approx$ # clauses for k-CNF)

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Refutation ends when empty clause $\perp$ derived

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Refutation ends when empty clause $\bot$ derived

1. $x \lor y$
2. $x \lor \overline{y} \lor z$
3. $\overline{x} \lor z$
4. $\overline{y} \lor \overline{z}$
5. $\overline{x} \lor \overline{z}$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \lor y$	Axiom
2.	$x \lor \overline{y} \lor z$	Axiom
3.	$\overline{x} \lor z$	Axiom
4.	$\overline{y} \lor \overline{z}$	Axiom
5.	$\overline{x} \lor \overline{z}$	Axiom
6.	$x \lor \overline{y}$	Res$(2, 4)$
7.	x	Res$(1, 6)$
8.	$\overline{x}$	Res$(3, 5)$
9.	$\bot$	Res$(7, 8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \lor y$	Axiom
2.	$\boldsymbol{x \lor \overline{y} \lor z}$	**Axiom**
3.	$\overline{x} \lor z$	Axiom
4.	$\boldsymbol{\overline{y} \lor \overline{z}}$	**Axiom**
5.	$\overline{x} \lor \overline{z}$	Axiom
6.	$x \lor \overline{y}$	Res$(2,4)$
7.	x	Res$(1,6)$
8.	$\overline{x}$	Res$(3,5)$
9.	$\bot$	Res$(7,8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \lor y$	Axiom
2.	$\boldsymbol{x \lor \overline{y} \lor z}$	Axiom
3.	$\overline{x} \lor z$	Axiom
4.	$\boldsymbol{\overline{y} \lor \overline{z}}$	Axiom
5.	$\overline{x} \lor \overline{z}$	Axiom
6.	$\boldsymbol{x \lor \overline{y}}$	Res$(2, 4)$
7.	x	Res$(1, 6)$
8.	$\overline{x}$	Res$(3, 5)$
9.	$\bot$	Res$(7, 8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \lor y$	Axiom
2.	$x \lor \overline{y} \lor z$	Axiom
3.	$\overline{x} \lor z$	Axiom
4.	$\overline{y} \lor \overline{z}$	Axiom
5.	$\overline{x} \lor \overline{z}$	Axiom
6.	$\boldsymbol{x \lor \overline{y}}$	Res$(2, 4)$
7.	x	Res$(1, 6)$
8.	$\overline{x}$	Res$(3, 5)$
9.	$\bot$	Res$(7, 8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$\boldsymbol{x \lor y}$	Axiom
2.	$x \lor \overline{y} \lor z$	Axiom
3.	$\overline{x} \lor z$	Axiom
4.	$\overline{y} \lor \overline{z}$	Axiom
5.	$\overline{x} \lor \overline{z}$	Axiom
6.	$\boldsymbol{x \lor \overline{y}}$	Res$(2,4)$
7.	x	Res$(1,6)$
8.	$\overline{x}$	Res$(3,5)$
9.	$\bot$	Res$(7,8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Refutation ends when empty clause $\perp$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$\boldsymbol{x \lor y}$	Axiom
2.	$x \lor \overline{y} \lor z$	Axiom
3.	$\overline{x} \lor z$	Axiom
4.	$\overline{y} \lor \overline{z}$	Axiom
5.	$\overline{x} \lor \overline{z}$	Axiom
6.	$\boldsymbol{x \lor \overline{y}}$	Res$(2,4)$
7.	$\boldsymbol{x}$	Res$(1,6)$
8.	$\overline{x}$	Res$(3,5)$
9.	$\perp$	Res$(7,8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \lor y$	Axiom
2.	$x \lor \overline{y} \lor z$	Axiom
3.	$\overline{x} \lor z$	Axiom
4.	$\overline{y} \lor \overline{z}$	Axiom
5.	$\overline{x} \lor \overline{z}$	Axiom
6.	$x \lor \overline{y}$	Res$(2,4)$
7.	$\boldsymbol{x}$	Res$(1,6)$
8.	$\overline{x}$	Res$(3,5)$
9.	$\bot$	Res$(7,8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \lor y$	Axiom
2.	$x \lor \overline{y} \lor z$	Axiom
3.	$\overline{x} \lor z$	Axiom
4.	$\overline{y} \lor \overline{z}$	Axiom
5.	$\overline{x} \lor \overline{z}$	Axiom
6.	$x \lor \overline{y}$	Res$(2, 4)$
7.	x	Res$(1, 6)$
8.	$\overline{x}$	Res$(3, 5)$
9.	$\bot$	Res$(7, 8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \lor y$	Axiom
2.	$x \lor \overline{y} \lor z$	Axiom
3.	$\overline{x} \lor z$	Axiom
4.	$\overline{y} \lor \overline{z}$	Axiom
5.	$\overline{x} \lor \overline{z}$	Axiom
6.	$x \lor \overline{y}$	Res$(2, 4)$
7.	x	Res$(1, 6)$
8.	$\overline{x}$	Res$(3, 5)$
9.	$\bot$	Res$(7, 8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \lor y$	Axiom
2.	$x \lor \overline{y} \lor z$	Axiom
3.	$\overline{x} \lor z$	Axiom
4.	$\overline{y} \lor \overline{z}$	Axiom
5.	$\overline{x} \lor \overline{z}$	Axiom
6.	$x \lor \overline{y}$	Res$(2, 4)$
7.	x	Res$(1, 6)$
8.	$\overline{x}$	Res$(3, 5)$
9.	$\bot$	Res$(7, 8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \vee y$	Axiom
2.	$x \vee \overline{y} \vee z$	Axiom
3.	$\overline{x} \vee z$	Axiom
4.	$\overline{y} \vee \overline{z}$	Axiom
5.	$\overline{x} \vee \overline{z}$	Axiom
6.	$x \vee \overline{y}$	Res$(2, 4)$
7.	$\boldsymbol{x}$	Res$(1, 6)$
8.	$\boldsymbol{\overline{x}}$	Res$(3, 5)$
9.	$\bot$	Res$(7, 8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \lor y$	Axiom
2.	$x \lor \overline{y} \lor z$	Axiom
3.	$\overline{x} \lor z$	Axiom
4.	$\overline{y} \lor \overline{z}$	Axiom
5.	$\overline{x} \lor \overline{z}$	Axiom
6.	$x \lor \overline{y}$	Res$(2,4)$
7.	$\boldsymbol{x}$	Res$(1,6)$
8.	$\overline{\boldsymbol{x}}$	Res$(3,5)$
9.	$\bot$	Res$(7,8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \lor y$	Axiom
2.	$x \lor \overline{y} \lor z$	Axiom
3.	$\overline{x} \lor z$	Axiom
4.	$\overline{y} \lor \overline{z}$	Axiom
5.	$\overline{x} \lor \overline{z}$	Axiom
6.	$x \lor \overline{y}$	Res$(2, 4)$
7.	x	Res$(1, 6)$
8.	$\overline{x}$	Res$(3, 5)$
9.	$\bot$	Res$(7, 8)$

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

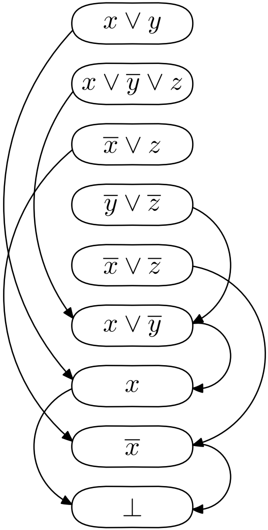
Derive new clauses by resolution rule

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

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph



The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

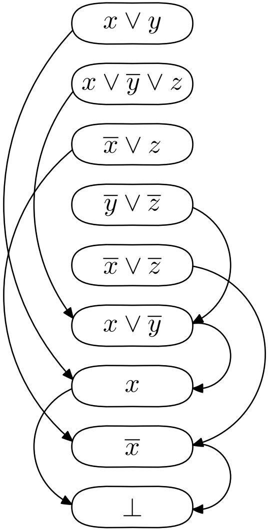
$$\frac{C \lor x \qquad D \lor \overline{x}}{C \lor D}$$

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

Tree-like resolution if DAG is tree



Resolution Size/Length

Size/length $= \#$ clauses in refutation

Most fundamental measure in proof complexity

Never worse than $\exp(\mathcal{O}(N))$

Matching $\exp(\Omega(N))$ lower bounds known

Examples of Hard Formulas w.r.t Resolution Size (1/2)

Pigeonhole principle (PHP) [Haken '85]
"$n + 1$ pigeons don't fit into n holes"

Variables $p_{i,j} =$ "pigeon i goes into hole j"

$p_{i,1} \lor p_{i,2} \lor \cdots \lor p_{i,n}$	every pigeon i gets a hole
$\overline{p}_{i,j} \lor \overline{p}_{i',j}$	no hole j gets two pigeons $i \neq i'$

Can also add "functionality" and "onto" axioms

$\overline{p}_{i,j} \lor \overline{p}_{i,j'}$	no pigeon i gets two holes $j \neq j'$
$p_{1,j} \lor p_{2,j} \lor \cdots \lor p_{n+1,j}$	every hole j gets a pigeon

Examples of Hard Formulas w.r.t Resolution Size (1/2)

Pigeonhole principle (PHP) [Haken '85]
"$n+1$ pigeons don't fit into n holes"

Variables $p_{i,j}$ = "pigeon i goes into hole j"

$p_{i,1} \lor p_{i,2} \lor \cdots \lor p_{i,n}$	every pigeon i gets a hole
$\overline{p}_{i,j} \lor \overline{p}_{i',j}$	no hole j gets two pigeons $i \neq i'$

Can also add "functionality" and "onto" axioms

$\overline{p}_{i,j} \lor \overline{p}_{i,j'}$	no pigeon i gets two holes $j \neq j'$
$p_{1,j} \lor p_{2,j} \lor \cdots \lor p_{n+1,j}$	every hole j gets a pigeon

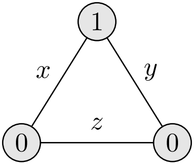
Even onto functional PHP formula is hard for resolution
"Resolution cannot count"

Examples of Hard Formulas w.r.t Resolution Size (2/2)

Tseitin formulas [Urquhart '87]
"Sum of degrees of vertices in graph is even"

Variables = edges (in undirected graph of bounded degree)

- Label every vertex 0/1 so that sum of labels odd
- Write CNF requiring parity of # true incident edges = label



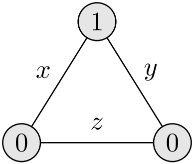
$$
\begin{aligned}
&(x \lor y) && \land\ (\overline{x} \lor z) \\
\land\ &(\overline{x} \lor \overline{y}) && \land\ (y \lor \overline{z}) \\
\land\ &(x \lor \overline{z}) && \land\ (\overline{y} \lor z)
\end{aligned}
$$

Examples of Hard Formulas w.r.t Resolution Size (2/2)

Tseitin formulas [Urquhart '87]
"Sum of degrees of vertices in graph is even"

Variables = edges (in undirected graph of bounded degree)

- Label every vertex 0/1 so that sum of labels odd
- Write CNF requiring parity of # true incident edges = label



$$
\begin{array}{ll}
(x \lor y) & \land\ (\overline{x} \lor z) \\
\land\ (\overline{x} \lor \overline{y}) & \land\ (y \lor \overline{z}) \\
\land\ (x \lor \overline{z}) & \land\ (\overline{y} \lor z)
\end{array}
$$

Requires size $\exp(\Omega(N))$ on well-connected so-called expanders
"Resolution cannot count $\mathrm{mod}\ 2$"

Resolution Width

Width = size of largest clause in refutation (always $\leq N$)

Resolution Width

Width $=$ size of largest clause in refutation (always $\leq N$)

Width upper bound $\Rightarrow$ size upper bound

Proof: at most $(2 \cdot \#\text{variables})^{\text{width}}$ distinct clauses
(This simple counting argument is essentially tight [Atserias et al.'14])

Resolution Width

Width $=$ size of largest clause in refutation (always $\leq N$)

Width upper bound $\Rightarrow$ size upper bound

Proof: at most $(2 \cdot \#\text{variables})^{\text{width}}$ distinct clauses
(This simple counting argument is essentially tight [Atserias et al.'14])

Width lower bound $\Rightarrow$ size lower bound

Much less obvious...

Width Lower Bounds Imply Size Lower Bounds

Theorem ([Ben-Sasson & Wigderson '99])

$$\mathit{size} \geq \exp\left(\Omega\left(\frac{(\mathit{width})^2}{(\mathit{formula\ size}\ N)}\right)\right)$$

Width Lower Bounds Imply Size Lower Bounds

Theorem ([Ben-Sasson & Wigderson '99])

$$\textit{size} \geq \exp\left(\Omega\left(\frac{(\textit{width})^2}{(\textit{formula size } N)}\right)\right)$$

Yields superpolynomial size bounds for width $\omega\left(\sqrt{N \log N}\right)$
Almost all known lower bounds on size derivable via width

Width Lower Bounds Imply Size Lower Bounds

Theorem ([Ben-Sasson & Wigderson '99])

$$\textit{size} \geq \exp\left(\Omega\left(\frac{(\textit{width})^2}{(\textit{formula size } N)}\right)\right)$$

Yields superpolynomial size bounds for width $\omega\left(\sqrt{N \log N}\right)$
Almost all known lower bounds on size derivable via width

For tree-like resolution have size $\geq 2^{\text{width}}$ [Ben-Sasson & Wigderson '99]

General resolution: width up to $\mathcal{O}\left(\sqrt{N \log N}\right)$ implies no size lower bounds — possible to tighten analysis? **No!**

Optimality of the Size-Width Lower Bound

Ordering principles [Stålmarck '96, Bonet & Galesi '99]
"Every (partially) ordered set $\{e_1, \ldots, e_n\}$ has minimal element"

Variables $x_{i,j} =$ "$e_i < e_j$"

$\overline{x}_{i,j} \vee \overline{x}_{j,i}$	anti-symmetry; not both $e_i < e_j$ and $e_j < e_i$
$\overline{x}_{i,j} \vee \overline{x}_{j,k} \vee x_{i,k}$	transitivity; $e_i < e_j$ and $e_j < e_k$ implies $e_i < e_k$
$\bigvee_{1 \leq i \leq n,\, i \neq j} x_{i,j}$	e_j is not a minimal element

Optimality of the Size-Width Lower Bound

Ordering principles [Stålmarck '96, Bonet & Galesi '99]
"Every (partially) ordered set $\{e_1, \ldots, e_n\}$ has minimal element"

Variables $x_{i,j} =$ "$e_i < e_j$"

$\overline{x}_{i,j} \vee \overline{x}_{j,i}$	anti-symmetry; not both $e_i < e_j$ and $e_j < e_i$
$\overline{x}_{i,j} \vee \overline{x}_{j,k} \vee x_{i,k}$	transitivity; $e_i < e_j$ and $e_j < e_k$ implies $e_i < e_k$
$\bigvee_{1 \leq i \leq n,\, i \neq j} x_{i,j}$	e_j is not a minimal element

Refutable in resolution in size $\mathcal{O}(N)$
Requires resolution width $\Omega\big(\sqrt[3]{N}\big)$ (converted to 3-CNF)

Polynomial Calculus

Introduced in [Clegg et al. '96]; slightly modified in [Alekhnovich et al. '00]

Clauses interpreted as polynomial equations over finite field
Any field in theory; $\mathrm{GF}(2)$ in practice
Example: $x \vee y \vee \overline{z}$ gets translated to $xy\overline{z} = 0$
(Think of $0 \equiv true$ and $1 \equiv false$)

Polynomial Calculus

Introduced in [Clegg et al. '96]; slightly modified in [Alekhnovich et al. '00]

Clauses interpreted as polynomial equations over finite field
Any field in theory; $\mathrm{GF}(2)$ in practice
Example: $x \lor y \lor \overline{z}$ gets translated to $xy\overline{z} = 0$
(Think of $0 \equiv \mathit{true}$ and $1 \equiv \mathit{false}$)

Derivation rules

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

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

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

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

Goal: Derive $1 = 0 \Leftrightarrow$ no common root $\Leftrightarrow$ formula unsatisfiable

Polynomial Calculus Size and Degree

Clauses turn into monomials
Write out all polynomials as sums of monomials
W.l.o.g. all polynomials multilinear (because of Boolean axioms)

Polynomial Calculus Size and Degree

Clauses turn into monomials
Write out all polynomials as sums of monomials
W.l.o.g. all polynomials multilinear (because of Boolean axioms)

Size — analogue of resolution length/size
total # monomials in refutation counted with repetitions

Degree — analogue of resolution width
largest degree of monomial in refutation

Polynomial Calculus Can Mimic Resolution Steps

Example: Resolution step:

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

Polynomial Calculus Can Mimic Resolution Steps

Example: Resolution step:

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

simulated by polynomial calculus derivation:

$$\dfrac{x\overline{y}z = 0 \qquad \dfrac{\dfrac{\overline{y}\overline{z} = 0}{x\overline{y}\overline{z} = 0} \qquad \dfrac{\dfrac{z + \overline{z} - 1 = 0}{\overline{y}z + \overline{y}\overline{z} - \overline{y} = 0}}{x\overline{y}z + x\overline{y}\overline{z} - x\overline{y} = 0}}{-x\overline{y}z + x\overline{y} = 0}}{x\overline{y} = 0}$$

Polynomial Calculus Strictly Stronger than Resolution

Polynomial calculus simulates resolution efficiently

- Can mimic refutation step by step as shown on previous slide
- Essentially no increase in length/size or width/degree
- Hence worst-case upper bounds for resolution carry over

Polynomial calculus is strictly stronger w.r.t. both size and degree

- Consider, e.g., Tseitin formulas on expanders
- Over $\mathrm{GF}(2)$ can just do Gaussian elimination
- Also other examples not depending on field characteristic

Size vs. Degree

- Degree upper bound ⇒ size upper bound [Clegg et al.'96]
  Qualitatively similar to resolution bound
  A bit more involved argument
  Again essentially tight by [Atserias et al.'14]

- Degree lower bound ⇒ size lower bound [Impagliazzo et al.'99]
  Precursor of [Ben-Sasson & Wigderson '99] — can do same proof to get same bound

- Size-degree lower bound essentially optimal [Galesi & Lauria '10]
  Example: same ordering principle formulas

- Most size lower bounds for polynomial calculus derived via degree lower bounds (but machinery much less developed)

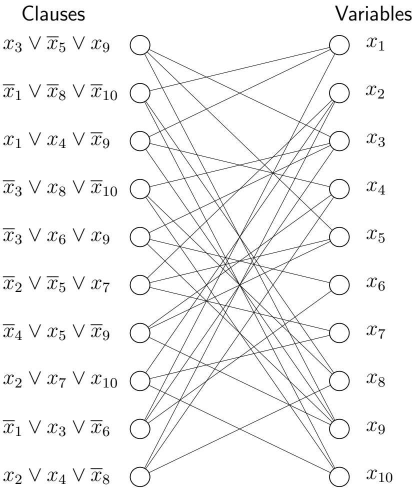
Lower Bounds via Expansion

Standard approach: Lower bounds from expansion

Simplest example: Clause-variable incidence graph (CVIG)

Standard approach: Lower bounds from expansion

Simplest example: Clause-variable incidence graph (CVIG)

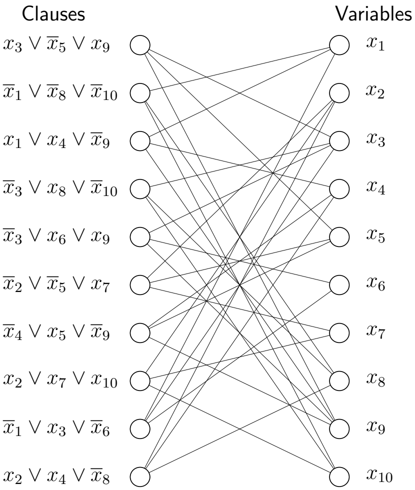


Lower Bounds via Expansion

Standard approach: Lower bounds from expansion

Simplest example: Clause-variable incidence graph (CVIG)

Boundary expansion:
Subsets of left vertices have many unique neighbours on right

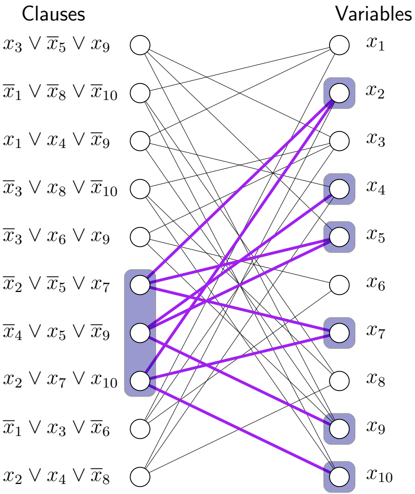


Lower Bounds via Expansion

Standard approach: Lower bounds from expansion

Simplest example: Clause-variable incidence graph (CVIG)

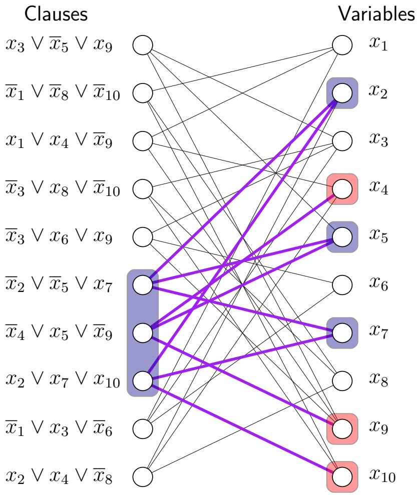
Boundary expansion:
Subsets of left vertices have many unique neighbours on right



Standard approach: Lower bounds from expansion

Simplest example: Clause-variable incidence graph (CVIG)

Boundary expansion:
Subsets of left vertices have many unique neighbours on right

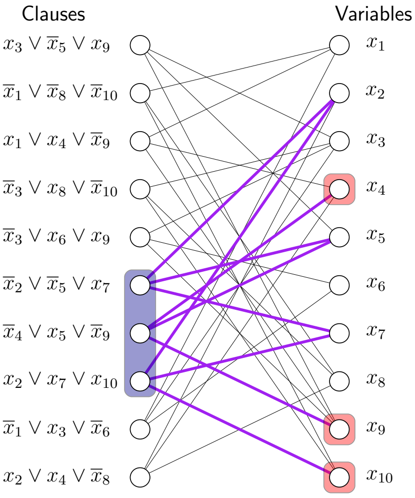


Lower Bounds via Expansion

Standard approach: Lower bounds from expansion

Simplest example: Clause-variable incidence graph (CVIG)

Boundary expansion:
Subsets of left vertices have many unique neighbours on right

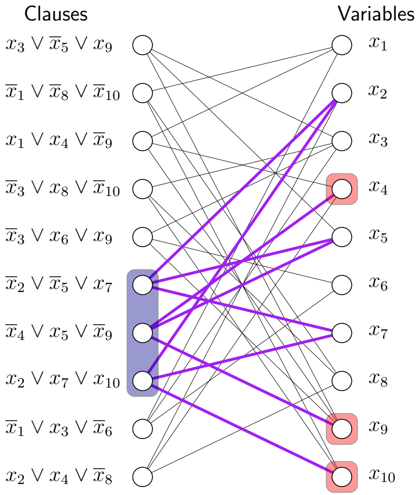


Standard approach: Lower bounds from expansion

Simplest example: Clause-variable incidence graph (CVIG)

Boundary expansion:
Subsets of left vertices have many unique neighbours on right

Problem:
CVIG might lose expansion of combinatorial problem



Lower Bounds via Expansion

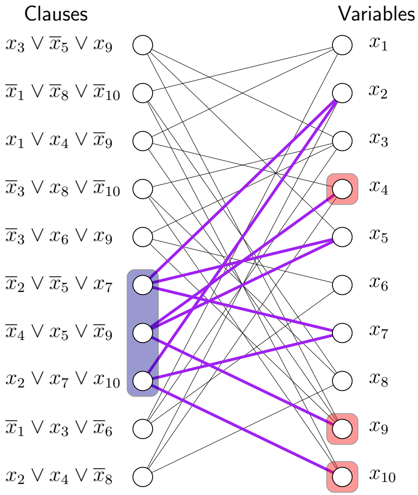
Standard approach: Lower bounds from expansion

Simplest example: Clause-variable incidence graph (CVIG)

Boundary expansion:
Subsets of left vertices have many unique neighbours on right

Problem:
CVIG might lose expansion of combinatorial problem

Need graph capturing underlying principle!



Our Results

Main Theorem (Informal)

Graph structure on formula such that expansion implies hardness in polynomial calculus

Extends an approach from [Alekhnovich, Razborov '01]

Unifies many previous lower bounds for polynomial calculus

Corollary: New lower bound resolving open question in [Razborov '02]

Our Results

Main Theorem (Informal)

Graph structure on formula such that expansion implies hardness in polynomial calculus

Extends an approach from [Alekhnovich, Razborov '01]

Unifies many previous lower bounds for polynomial calculus

Corollary: New lower bound resolving open question in [Razborov '02]

Warm-up: Use resolution to present main ideas and challenges

Revisiting Tseitin Formulas

Given set of equations over $\mathbb{F}_2$

$$\begin{aligned} x + w &= 0 \\ x + y &= 0 \\ y + w + z &= 1 \\ z &= 0 \end{aligned}$$

Revisiting Tseitin Formulas

Given set of equations over $\mathbb{F}_2$

$$x + w = 0$$
$$x + y = 0$$
$$y + w + z = 1$$
$$z = 0$$

Encode as clauses

Clauses

$x \lor \overline{w}$

$\overline{x} \lor w$

$x \lor \overline{y}$

$\overline{x} \lor y$

$y \lor w \lor z$

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

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

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

$\overline{z}$

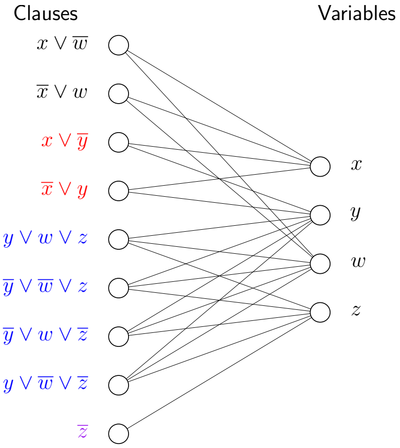
Revisiting Tseitin Formulas

Given set of equations over $\mathbb{F}_2$

$$x + w = 0$$
$$x + y = 0$$
$$y + w + z = 1$$
$$z = 0$$

Encode as clauses

Does CVIG expand?



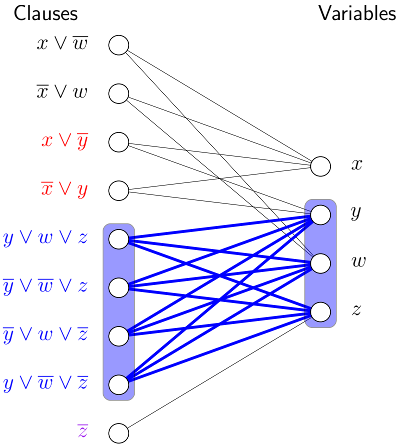
Revisiting Tseitin Formulas

Given set of equations over $\mathbb{F}_2$

$$x + w = 0$$
$$x + y = 0$$
$$y + w + z = 1$$
$$z = 0$$

Encode as clauses

Does CVIG expand? No!



Revisiting Tseitin Formulas

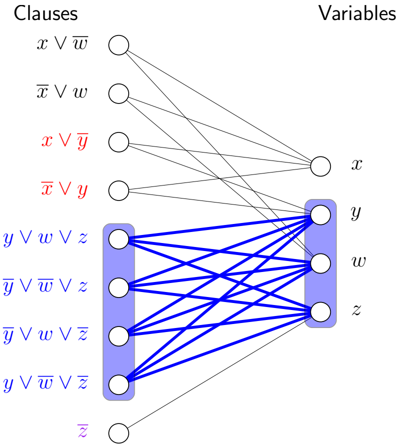
Given set of equations over $\mathbb{F}_2$

$$x + w = 0$$
$$x + y = 0$$
$$y + w + z = 1$$
$$z = 0$$

Encode as clauses

Does CVIG expand? No!

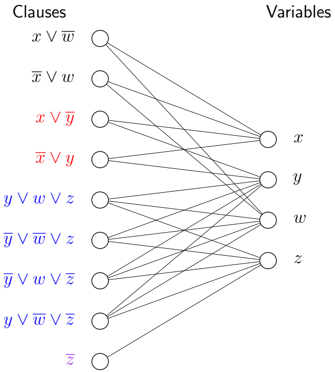
Graph should encode equations, not clauses!

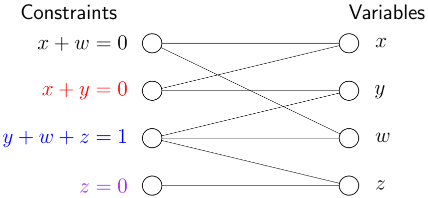


Constraint-Variable Incidence Graph

Use one vertex per equation on the left

Put edge if variable appears in equation

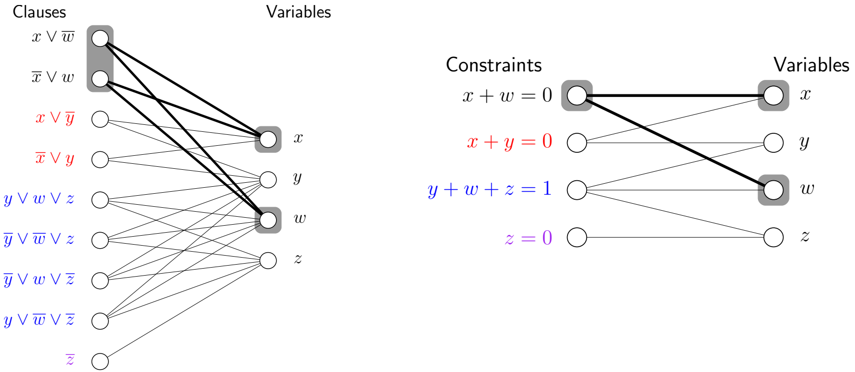




Constraint-Variable Incidence Graph

Use one vertex per equation on the left

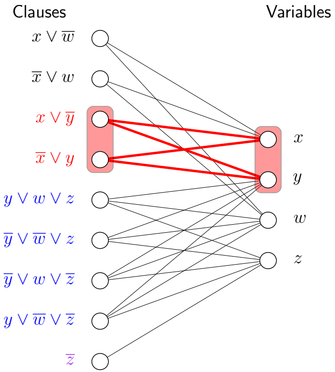
Put edge if variable appears in equation

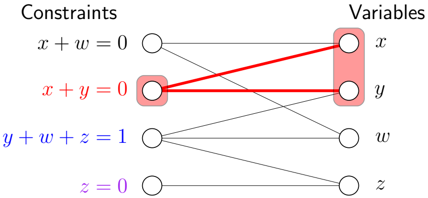


Constraint-Variable Incidence Graph

Use one vertex per equation on the left

Put edge if variable appears in equation

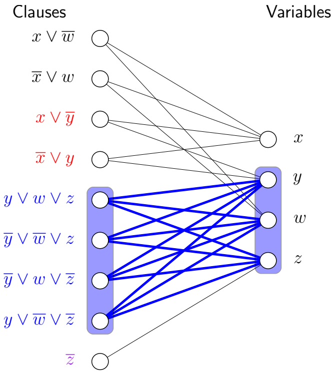


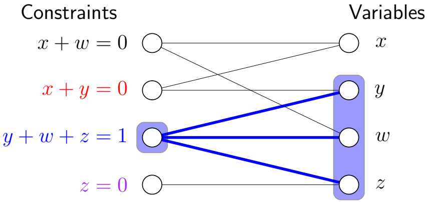


Constraint-Variable Incidence Graph

Use one vertex per equation on the left

Put edge if variable appears in equation

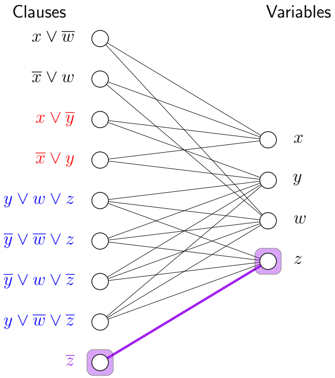


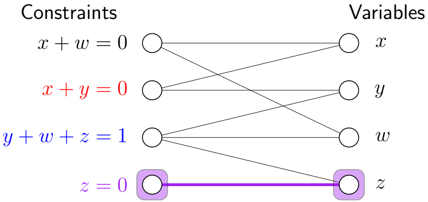


Constraint-Variable Incidence Graph

Use one vertex per equation on the left

Put edge if variable appears in equation

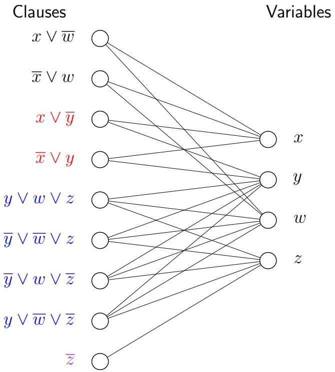


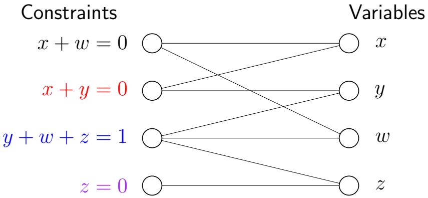


Constraint-Variable Incidence Graph

Use one vertex per equation on the left

Put edge if variable appears in equation

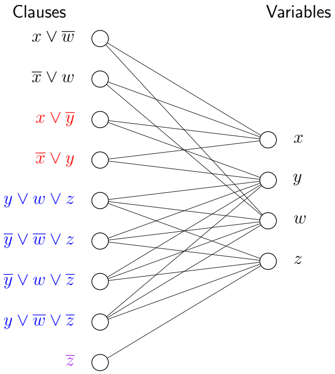


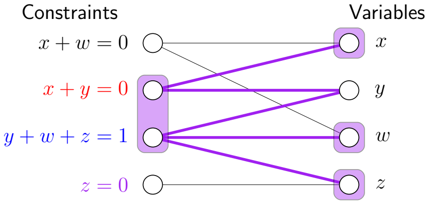


Constraint-Variable Incidence Graph

Use one vertex per equation on the left

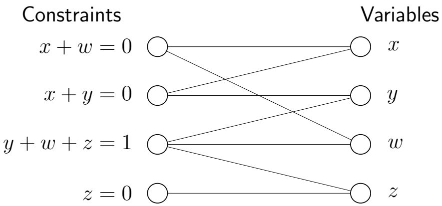
Put edge if variable appears in equation



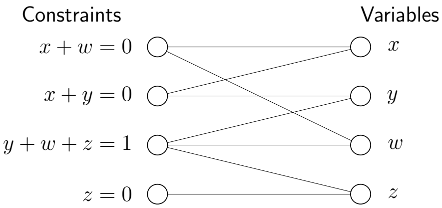


Now the constraint-variable incidence graph expands!

Proof Sketch of Tseitin Lower Bound

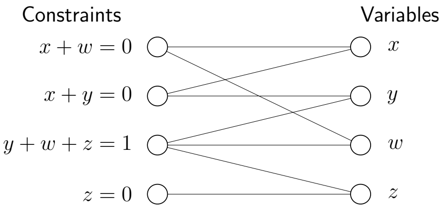


Proof Sketch of Tseitin Lower Bound



1. For each clause, look at minimal set of constraints implying it

Proof Sketch of Tseitin Lower Bound

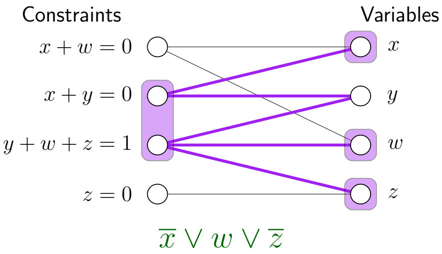


1. For each clause, look at minimal set of constraints implying it

   Axioms: 1 constraint needed

   Contradiction $\perp$: All constraints needed

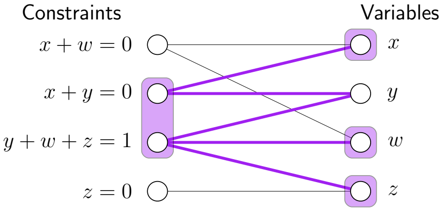
Proof Sketch of Tseitin Lower Bound



1. For each clause, look at minimal set of constraints implying it

Axioms:	1 constraint needed
Contradiction $\perp$:	All constraints needed
Halfway through:	Clause C depending on medium-sized set S

Proof Sketch of Tseitin Lower Bound



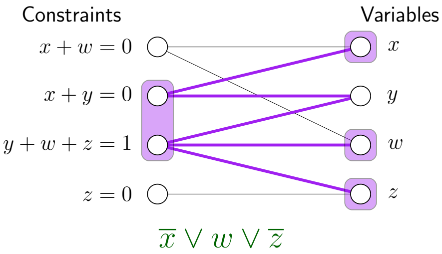
$\overline{x} \vee w \vee \overline{z}$

1. For each clause, look at minimal set of constraints implying it

Axioms:	1 constraint needed
Contradiction $\perp$:	All constraints needed
Halfway through:	Clause C depending on medium-sized set S

2. S has large boundary expansion $\Rightarrow$ All boundary variables in C

Proof Sketch of Tseitin Lower Bound

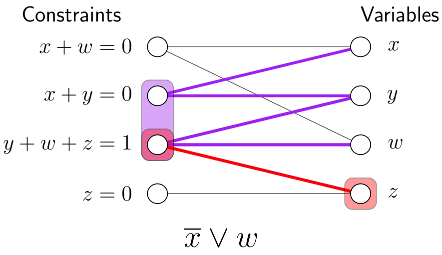


1. For each clause, look at minimal set of constraints implying it

Axioms:	1 constraint needed
Contradiction $\perp$:	All constraints needed
Halfway through:	Clause C depending on medium-sized set S

2. S has large boundary expansion $\Rightarrow$ All boundary variables in C
3. Proof: Suppose not $\Rightarrow$ not all of S needed for C

Proof Sketch of Tseitin Lower Bound

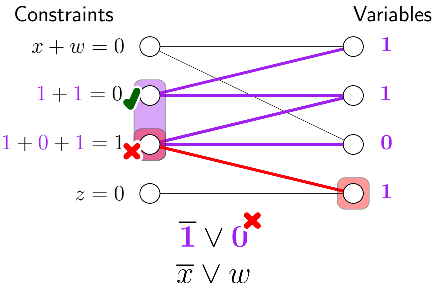


1. For each clause, look at minimal set of constraints implying it

Axioms:	1 constraint needed
Contradiction $\perp$:	All constraints needed
Halfway through:	Clause C depending on medium-sized set S

2. S has large boundary expansion $\Rightarrow$ All boundary variables in C
3. Proof: Suppose not $\Rightarrow$ not all of S needed for C

Proof Sketch of Tseitin Lower Bound

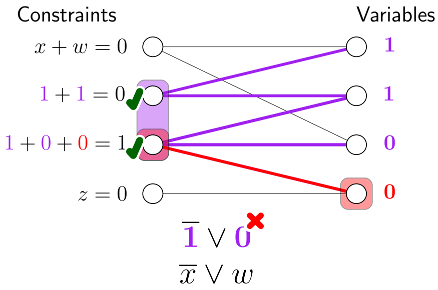


1. For each clause, look at minimal set of constraints implying it

Axioms:	1 constraint needed
Contradiction $\perp$:	All constraints needed
Halfway through:	Clause C depending on medium-sized set S

2. S has large boundary expansion $\Rightarrow$ All boundary variables in C
3. Proof: Suppose not $\Rightarrow$ not all of S needed for C

Proof Sketch of Tseitin Lower Bound



1. For each clause, look at minimal set of constraints implying it

Axioms:	1 constraint needed
Contradiction $\perp$:	All constraints needed
Halfway through:	Clause C depending on medium-sized set S

2. S has large boundary expansion $\Rightarrow$ All boundary variables in C
3. Proof: Suppose not $\Rightarrow$ not all of S needed for C

Resolution Lower Bounds

Resolution edge game on (P, x)

1. Adversary provides assignment ρ to all variables
2. Can flip x to some b so that P is satisfied

Resolution Lower Bounds

Resolution edge game on (P, x)

1. Adversary provides assignment ρ to all variables
2. Can flip x to some b so that P is satisfied

Theorem (Ben-Sasson & Wigderson '99)

If from formula $\mathcal{F} = \bigwedge_{P \in \mathcal{F}} P$ can form bipartite graph $\mathcal{G}(\mathcal{F})$ such that

- *$\mathcal{G}(\mathcal{F})$ is expanding and*
- *for all edges (P, x) we can satisfy P by flipping x*

then refuting $\mathcal{F}$ requires large width

Polynomial Calculus Edge Game

Tseitin: linear equations $\Rightarrow$ easy over $\mathbb{F}_2$ (Gaussian elimination)

Need stronger guarantee from constraint-variable incidence graph!

Polynomial Calculus Edge Game

Tseitin: linear equations $\Rightarrow$ easy over $\mathbb{F}_2$ (Gaussian elimination)

Need stronger guarantee from constraint-variable incidence graph!

Resolution graph:

- Graph is boundary expander
- Can play resolution edge game on every edge (P, x)

Polynomial Calculus Edge Game

Tseitin: linear equations $\Rightarrow$ easy over $\mathbb{F}_2$ (Gaussian elimination)

Need stronger guarantee from constraint-variable incidence graph!

Resolution graph:

- Graph is boundary expander
- Can play resolution edge game on every edge (P, x)

For polynomial calculus we have to play a harder game

Polynomial Calculus Edge Game

Tseitin: linear equations $\Rightarrow$ easy over $\mathbb{F}_2$ (Gaussian elimination)

Need stronger guarantee from constraint-variable incidence graph!

Resolution graph:

- Graph is boundary expander
- Can play resolution edge game on every edge (P, x)

For polynomial calculus we have to play a harder game

Polynomial calculus edge game on (P, x)

1. Commit to assignment $x = b$ ahead of time
2. Adversary provides assignment ρ to all variables
3. Flipping $x = b$ satisfies P

Easy to see we can't win this game for Tseitin formulas

Main Theorem (Preliminary Version)

If from formula $\mathcal{F} = \bigwedge_{P \in \mathcal{F}} P$ can form bipartite graph $\mathcal{G}(\mathcal{F})$ such that:

- $\mathcal{G}(\mathcal{F})$ is expanding, and
- for all edges (P, x) can fix P to true by flipping x,

then refuting $\mathcal{F}$ requires large degree

Main Theorem (Preliminary Version)

If from formula $\mathcal{F} = \bigwedge_{P \in \mathcal{F}} P$ can form bipartite graph $\mathcal{G}(\mathcal{F})$ such that:

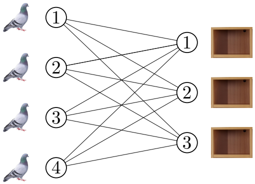
- $\mathcal{G}(\mathcal{F})$ is expanding, and
- for all edges (P, x) can fix P to true by flipping x,

then refuting $\mathcal{F}$ requires large degree

Not enough to prove functional pigeonhole principle hard!

Pigeonhole Principle (PHP)

"$n + 1$ pigeons don't fit into n holes"



Pigeonhole Principle (PHP)

"$n + 1$ pigeons don't fit into n holes"

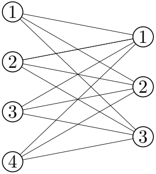
Pigeons

$x_{1,1} \lor x_{1,2} \lor x_{1,3}$

$x_{2,1} \lor x_{2,2} \lor x_{2,3}$

$x_{3,1} \lor x_{3,2} \lor x_{3,3}$

$x_{4,1} \lor x_{4,2} \lor x_{4,3}$



Holes

$\overline{x}_{1,1} \lor \overline{x}_{2,1}$ $\quad$ $\overline{x}_{2,1} \lor \overline{x}_{3,1}$
$\overline{x}_{1,1} \lor \overline{x}_{3,1}$ $\quad$ $\overline{x}_{2,1} \lor \overline{x}_{4,1}$
$\overline{x}_{1,1} \lor \overline{x}_{4,1}$ $\quad$ $\overline{x}_{3,1} \lor \overline{x}_{4,1}$

$\overline{x}_{1,2} \lor \overline{x}_{2,2}$ $\quad$ $\overline{x}_{2,2} \lor \overline{x}_{3,2}$
$\overline{x}_{1,2} \lor \overline{x}_{3,2}$ $\quad$ $\overline{x}_{2,2} \lor \overline{x}_{4,2}$
$\overline{x}_{1,2} \lor \overline{x}_{4,2}$ $\quad$ $\overline{x}_{3,2} \lor \overline{x}_{4,2}$

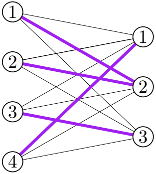
$\overline{x}_{1,3} \lor \overline{x}_{2,3}$ $\quad$ $\overline{x}_{2,3} \lor \overline{x}_{3,3}$
$\overline{x}_{1,3} \lor \overline{x}_{3,3}$ $\quad$ $\overline{x}_{2,3} \lor \overline{x}_{4,3}$
$\overline{x}_{1,3} \lor \overline{x}_{4,3}$ $\quad$ $\overline{x}_{3,3} \lor \overline{x}_{4,3}$

Pigeonhole Principle (PHP)

"$n+1$ pigeons don't fit into n holes"
Very wide clauses — hit with restriction to decrease width
Restricts choices of holes for each pigeon — graph PHP formula

Pigeons

$x_{1,1} \lor x_{1,2} \lor x_{1,3}$
$x_{2,1} \lor x_{2,2} \lor x_{2,3}$
$x_{3,1} \lor x_{3,2} \lor x_{3,3}$
$x_{4,1} \lor x_{4,2} \lor x_{4,3}$



Holes

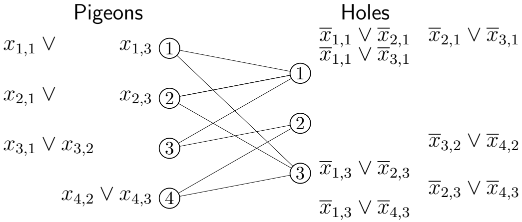
$\overline{x}_{1,1} \lor \overline{x}_{2,1}$ $\quad$ $\overline{x}_{2,1} \lor \overline{x}_{3,1}$
$\overline{x}_{1,1} \lor \overline{x}_{3,1}$ $\quad$ $\overline{x}_{2,1} \lor \overline{x}_{4,1}$
$\overline{x}_{1,1} \lor \overline{x}_{4,1}$ $\quad$ $\overline{x}_{3,1} \lor \overline{x}_{4,1}$

$\overline{x}_{1,2} \lor \overline{x}_{2,2}$ $\quad$ $\overline{x}_{2,2} \lor \overline{x}_{3,2}$
$\overline{x}_{1,2} \lor \overline{x}_{3,2}$ $\quad$ $\overline{x}_{2,2} \lor \overline{x}_{4,2}$
$\overline{x}_{1,2} \lor \overline{x}_{4,2}$ $\quad$ $\overline{x}_{3,2} \lor \overline{x}_{4,2}$

$\overline{x}_{1,3} \lor \overline{x}_{2,3}$ $\quad$ $\overline{x}_{2,3} \lor \overline{x}_{3,3}$
$\overline{x}_{1,3} \lor \overline{x}_{3,3}$ $\quad$ $\overline{x}_{2,3} \lor \overline{x}_{4,3}$
$\overline{x}_{1,3} \lor \overline{x}_{4,3}$ $\quad$ $\overline{x}_{3,3} \lor \overline{x}_{4,3}$

Pigeonhole Principle (PHP)

"$n + 1$ pigeons don't fit into n holes"
Very wide clauses — hit with restriction to decrease width
Restricts choices of holes for each pigeon — graph PHP formula

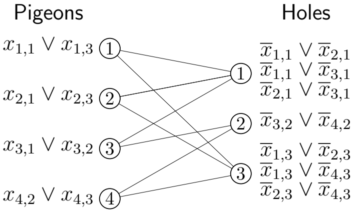


Pigeonhole Principle (PHP)

"$n+1$ pigeons don't fit into n holes"

Very wide clauses — hit with restriction to decrease width

Restricts choices of holes for each pigeon — graph PHP formula

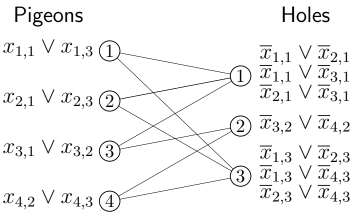


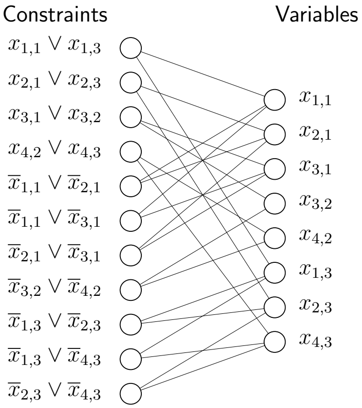
Pigeonhole Principle (PHP)

"$n + 1$ pigeons don't fit into n holes"

Very wide clauses — hit with restriction to decrease width

Restricts choices of holes for each pigeon — graph PHP formula



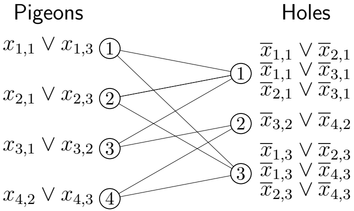


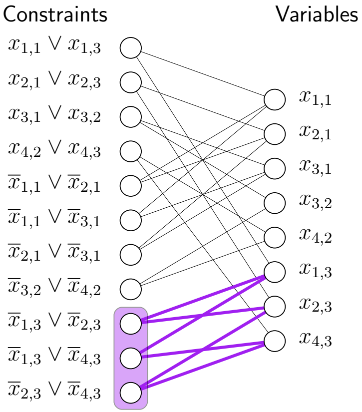
Pigeonhole Principle (PHP)

"$n+1$ pigeons don't fit into n holes"

Very wide clauses — hit with restriction to decrease width

Restricts choices of holes for each pigeon — graph PHP formula





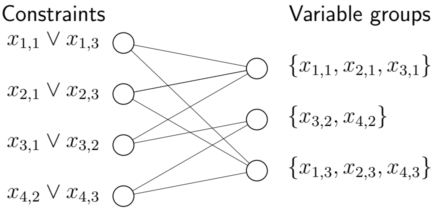
But again CVIG not expanding!

Proving PHP Lower Bound

Isolate hole axioms from graph and group hole variables together

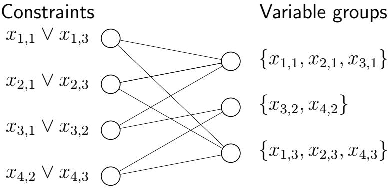
Proving PHP Lower Bound

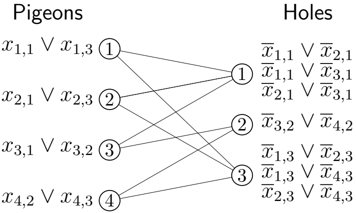
Isolate hole axioms from graph and group hole variables together



Proving PHP Lower Bound

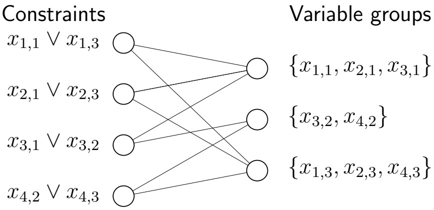
Isolate hole axioms from graph and group hole variables together





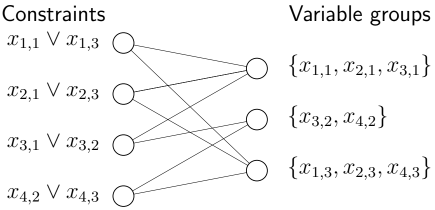
Proving PHP Lower Bound

Isolate hole axioms from graph and group hole variables together



Proving PHP Lower Bound

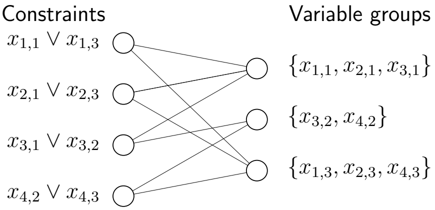
Isolate hole axioms from graph and group hole variables together



- Change game to play on assignments to groups of variables
- Assignments must satisfy any hole axioms touched

Proving PHP Lower Bound

Isolate hole axioms from graph and group hole variables together



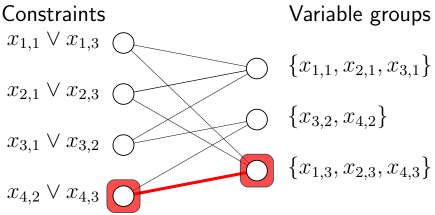
- Change game to play on assignments to groups of variables
- Assignments must satisfy any hole axioms touched

Polynomial calculus edge game on (P, V) with side constraint E

1. Commit to $\rho_V : V \to \{0, 1\}$ satisfying any clauses touched in E
2. Adversary provides total assignment ρ not violating E
3. Flipping V to ρ_V should now satisfy $P \wedge E$

Proving PHP Lower Bound

Isolate hole axioms from graph and group hole variables together



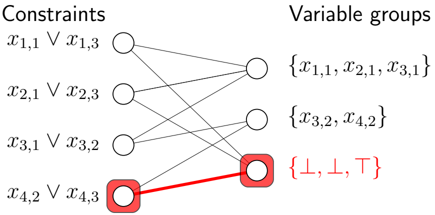
- Change game to play on assignments to groups of variables
- Assignments must satisfy any hole axioms touched

Polynomial calculus edge game on (P, V) with side constraint E

1. Commit to $\rho_V : V \to \{0, 1\}$ satisfying any clauses touched in E
2. Adversary provides total assignment ρ not violating E
3. Flipping V to ρ_V should now satisfy $P \wedge E$

Proving PHP Lower Bound

Isolate hole axioms from graph and group hole variables together



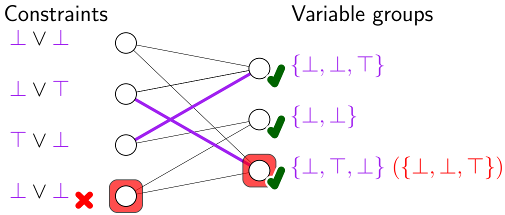
- Change game to play on assignments to groups of variables
- Assignments must satisfy any hole axioms touched

Polynomial calculus edge game on (P, V) with side constraint E

1. Commit to $\rho_V : V \to \{0, 1\}$ satisfying any clauses touched in E
2. Adversary provides total assignment ρ not violating E
3. Flipping V to ρ_V should now satisfy $P \wedge E$

Proving PHP Lower Bound

Isolate hole axioms from graph and group hole variables together



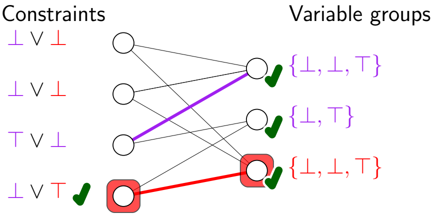
- Change game to play on assignments to groups of variables
- Assignments must satisfy any hole axioms touched

Polynomial calculus edge game on (P, V) with side constraint E

1. Commit to $\rho_V : V \to \{0, 1\}$ satisfying any clauses touched in E
2. Adversary provides total assignment ρ not violating E
3. Flipping V to ρ_V should now satisfy $P \wedge E$

Proving PHP Lower Bound

Isolate hole axioms from graph and group hole variables together



- Change game to play on assignments to groups of variables
- Assignments must satisfy any hole axioms touched

Polynomial calculus edge game on (P, V) with side constraint E

1. Commit to $\rho_V : V \to \{0, 1\}$ satisfying any clauses touched in E
2. Adversary provides total assignment ρ not violating E
3. Flipping V to ρ_V should now satisfy $P \wedge E$

Generalized Method for Degree Lower Bounds

Main Theorem

If from formula $\mathcal{F} = \mathcal{F}' \wedge E$ we can form bipartite graph $\mathcal{G}(\mathcal{F}')$ such that:

- $\mathcal{G}(\mathcal{F}')$ is expanding, and
- For all edges (P, V) there is an assignment to V
  - fixing P to true and
  - satisfying any touched clause in E

then refuting $\mathcal{F}$ requires large degree

Generalized Method for Degree Lower Bounds

Main Theorem

If from formula $\mathcal{F} = \mathcal{F}' \wedge E$ we can form bipartite graph $\mathcal{G}(\mathcal{F}')$ such that:

- $\mathcal{G}(\mathcal{F}')$ is expanding, and
- For all edges (P, V) there is an assignment to V
  - fixing P to true and
  - satisfying any touched clause in E

then refuting $\mathcal{F}$ requires large degree

Provides common framework for previous lower bounds:

- CNF formulas with expanding CVIGs [Alekhnovich & Razborov '01]
- Pigeonhole principle [Alekhnovich & Razborov '01]
- Graph ordering principle [Galesi & Lauria '10]

Generalized Method for Degree Lower Bounds

Main Theorem

If from formula $\mathcal{F} = \mathcal{F}' \wedge E$ we can form bipartite graph $\mathcal{G}(\mathcal{F}')$ such that:

- $\mathcal{G}(\mathcal{F}')$ is expanding, and
- For all edges (P, V) there is an assignment to V
  - fixing P to true and
  - satisfying any touched clause in E

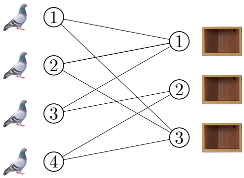
then refuting $\mathcal{F}$ requires large degree

Provides common framework for previous lower bounds:

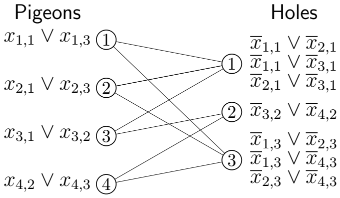
- CNF formulas with expanding CVIGs [Alekhnovich & Razborov '01]
- Pigeonhole principle [Alekhnovich & Razborov '01]
- Graph ordering principle [Galesi & Lauria '10]

Allows us to establish that functional PHP is hard

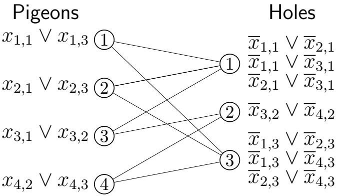
PHP Variants



PHP Variants

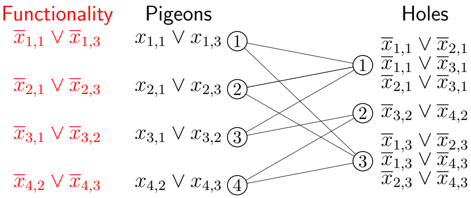


PHP Variants



- Can have “fat pigeons” assigned to multiple holes

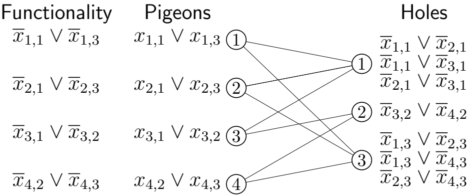
PHP Variants



- Can have "fat pigeons" assigned to multiple holes

  $\Rightarrow$ Add functionality axioms (makes mapping 1-to-1)

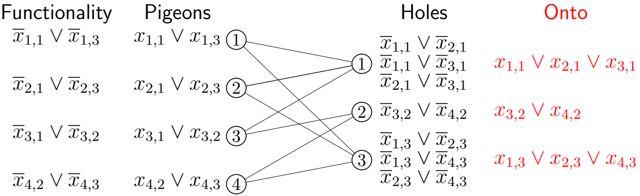
PHP Variants



- Can have "fat pigeons" assigned to multiple holes

  $\Rightarrow$ Add functionality axioms (makes mapping 1-to-1)
- Can have holes with no pigeons

PHP Variants

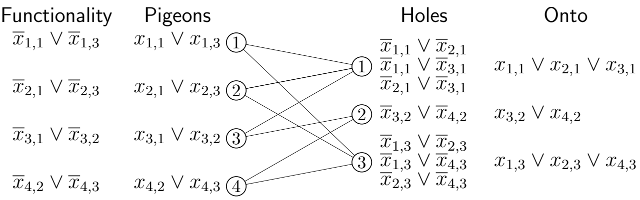


- Can have “fat pigeons” assigned to multiple holes

  $\Rightarrow$ Add functionality axioms (makes mapping 1-to-1)
- Can have holes with no pigeons

  $\Rightarrow$ Add onto axioms (makes mapping onto)

PHP Variants



- Can have "fat pigeons" assigned to multiple holes

  $\Rightarrow$ Add functionality axioms (makes mapping 1-to-1)

- Can have holes with no pigeons

  $\Rightarrow$ Add onto axioms (makes mapping onto)

PHP Variants

Functionality

$\overline{x}_{1,1} \lor \overline{x}_{1,3}$

$\overline{x}_{2,1} \lor \overline{x}_{2,3}$

$\overline{x}_{3,1} \lor \overline{x}_{3,2}$

$\overline{x}_{4,2} \lor \overline{x}_{4,3}$

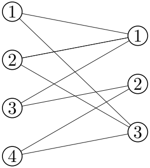
Pigeons

$x_{1,1} \lor x_{1,3}$

$x_{2,1} \lor x_{2,3}$

$x_{3,1} \lor x_{3,2}$

$x_{4,2} \lor x_{4,3}$



Holes

$\overline{x}_{1,1} \lor \overline{x}_{2,1}$
$\overline{x}_{1,1} \lor \overline{x}_{3,1}$
$\overline{x}_{2,1} \lor \overline{x}_{3,1}$

$\overline{x}_{3,2} \lor \overline{x}_{4,2}$

$\overline{x}_{1,3} \lor \overline{x}_{2,3}$
$\overline{x}_{1,3} \lor \overline{x}_{4,3}$
$\overline{x}_{2,3} \lor \overline{x}_{4,3}$

Onto

$x_{1,1} \lor x_{2,1} \lor x_{3,1}$

$x_{3,2} \lor x_{4,2}$

$x_{1,3} \lor x_{2,3} \lor x_{4,3}$

- Can have “fat pigeons” assigned to multiple holes

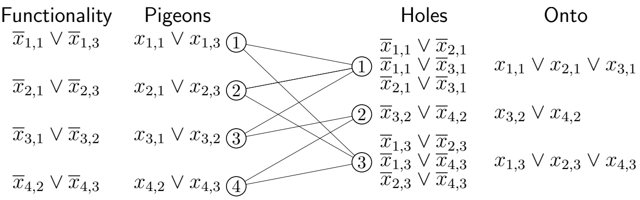
  $\Rightarrow$ Add functionality axioms (makes mapping 1-to-1)

- Can have holes with no pigeons

  $\Rightarrow$ Add onto axioms (makes mapping onto)

Functional PHP = PHP + Functionality

PHP Variants



- Can have "fat pigeons" assigned to multiple holes

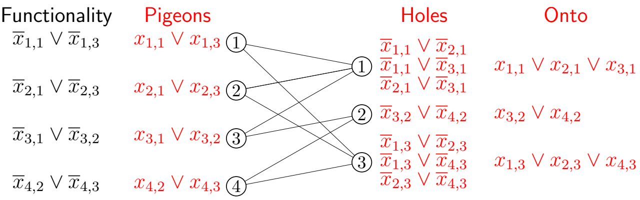
  $\Rightarrow$ Add functionality axioms (makes mapping 1-to-1)

- Can have holes with no pigeons

  $\Rightarrow$ Add onto axioms (makes mapping onto)

Functional PHP $=$ PHP $+$ Functionality

PHP Variants



- Can have “fat pigeons” assigned to multiple holes

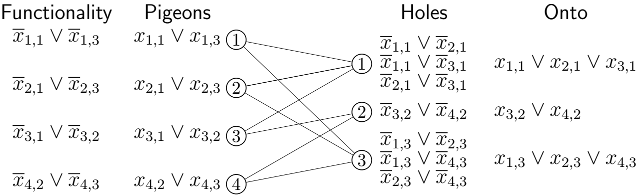
  $\Rightarrow$ Add functionality axioms (makes mapping 1-to-1)
- Can have holes with no pigeons

  $\Rightarrow$ Add onto axioms (makes mapping onto)

Functional PHP $=$ PHP $+$ Functionality

Onto-PHP $=$ PHP $+$ Onto

PHP Variants



- Can have "fat pigeons" assigned to multiple holes

  $\Rightarrow$ Add functionality axioms (makes mapping 1-to-1)

- Can have holes with no pigeons

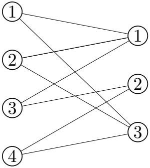
  $\Rightarrow$ Add onto axioms (makes mapping onto)

Functional PHP = PHP + Functionality

Onto-PHP = PHP + Onto

PHP Variants

Functionality	Pigeons
$\overline{x}_{1,1} \lor \overline{x}_{1,3}$	$x_{1,1} \lor x_{1,3}$
$\overline{x}_{2,1} \lor \overline{x}_{2,3}$	$x_{2,1} \lor x_{2,3}$
$\overline{x}_{3,1} \lor \overline{x}_{3,2}$	$x_{3,1} \lor x_{3,2}$
$\overline{x}_{4,2} \lor \overline{x}_{4,3}$	$x_{4,2} \lor x_{4,3}$



Holes	Onto
$\overline{x}_{1,1} \lor \overline{x}_{2,1}$ $\overline{x}_{1,1} \lor \overline{x}_{3,1}$ $\overline{x}_{2,1} \lor \overline{x}_{3,1}$	$x_{1,1} \lor x_{2,1} \lor x_{3,1}$
$\overline{x}_{3,2} \lor \overline{x}_{4,2}$	$x_{3,2} \lor x_{4,2}$
$\overline{x}_{1,3} \lor \overline{x}_{2,3}$ $\overline{x}_{1,3} \lor \overline{x}_{4,3}$ $\overline{x}_{2,3} \lor \overline{x}_{4,3}$	$x_{1,3} \lor x_{2,3} \lor x_{4,3}$

- Can have "fat pigeons" assigned to multiple holes

  $\Rightarrow$ Add functionality axioms (makes mapping 1-to-1)

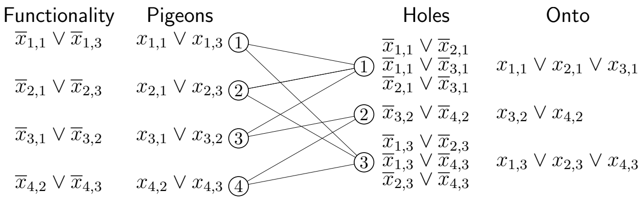
- Can have holes with no pigeons

  $\Rightarrow$ Add onto axioms (makes mapping onto)

Functional PHP = PHP + Functionality

Onto-PHP = PHP + Onto

Onto-FPHP = PHP + Functionality + Onto



- Can have "fat pigeons" assigned to multiple holes
  ⇒ Add functionality axioms (makes mapping 1-to-1)
- Can have holes with no pigeons
  ⇒ Add onto axioms (makes mapping onto)

Functional PHP = PHP + Functionality

Onto-PHP = PHP + Onto

Onto-FPHP = PHP + Functionality + Onto

Hardness of PHP Variants

Variant	Resolution	Polynomial calculus
PHP		
FPHP		
Onto-PHP		
Onto-FPHP		

Hardness of PHP Variants

Variant	Resolution	Polynomial calculus
PHP	hard [Haken '85]	
FPHP		
Onto-PHP		
Onto-FPHP		

Hardness of PHP Variants

Variant	Resolution	Polynomial calculus
PHP	hard [Haken '85]	
FPHP	hard [Haken '85]	
Onto-PHP	hard [Haken '85]	
Onto-FPHP	hard [Haken '85]	

Hardness of PHP Variants

Variant	Resolution	Polynomial calculus
PHP	hard [Haken '85]	hard [AR '01]
FPHP	hard [Haken '85]	
Onto-PHP	hard [Haken '85]	
Onto-FPHP	hard [Haken '85]	

Hardness of PHP Variants

Variant	Resolution	Polynomial calculus
PHP	hard [Haken '85]	hard [AR '01]
FPHP	hard [Haken '85]	
Onto-PHP	hard [Haken '85]	
Onto-FPHP	hard [Haken '85]	easy! [Riis '93]

Hardness of PHP Variants

Variant	Resolution	Polynomial calculus
PHP	hard [Haken '85]	hard [AR '01]
FPHP	hard [Haken '85]	?
Onto-PHP	hard [Haken '85]	?
Onto-FPHP	hard [Haken '85]	easy! [Riis '93]

Hardness of PHP Variants

Variant	Resolution	Polynomial calculus
PHP	hard [Haken '85]	hard [AR '01]
FPHP	hard [Haken '85]	?
Onto-PHP	hard [Haken '85]	hard [AR '01]
Onto-FPHP	hard [Haken '85]	easy! [Riis '93]

This work

- Observe that [AR '01] proves hardness of Onto-PHP

Hardness of PHP Variants

Variant	Resolution	Polynomial calculus
PHP	hard [Haken '85]	hard [AR '01]
FPHP	hard [Haken '85]	hard [MN '15]
Onto-PHP	hard [Haken '85]	hard [AR '01]
Onto-FPHP	hard [Haken '85]	easy! [Riis '93]

This work

- Observe that [AR '01] proves hardness of Onto-PHP
- Prove that FPHP is hard in polynomial calculus

Open Problems

- Prove polynomial calculus lower bounds for other formulas

Open Problems

- Prove polynomial calculus lower bounds for other formulas
  - graph colouring formulas
  - independent set formulas

Open Problems

- Prove polynomial calculus lower bounds for other formulas
  - graph colouring formulas
  - independent set formulas
- Prove size lower bounds via technique that doesn't use degree

Open Problems

- Prove polynomial calculus lower bounds for other formulas
  - graph colouring formulas
  - independent set formulas
- Prove size lower bounds via technique that doesn't use degree
  - clique formulas
  - weak pigeonhole principle formulas

Open Problems

- Prove polynomial calculus lower bounds for other formulas
  - graph colouring formulas
  - independent set formulas
- Prove size lower bounds via technique that doesn't use degree
  - clique formulas
  - weak pigeonhole principle formulas
- Find truly general framework capturing all PC degree lower bounds

Open Problems

- Prove polynomial calculus lower bounds for other formulas
  - graph colouring formulas
  - independent set formulas
- Prove size lower bounds via technique that doesn't use degree
  - clique formulas
  - weak pigeonhole principle formulas
- Find truly general framework capturing all PC degree lower bounds
  - We generalize only part of [Alekhnovich & Razborov '01]
  - Cannot deal with lower bounds à la [Buss et al. '99]

Open Problems

- Prove polynomial calculus lower bounds for other formulas
  - graph colouring formulas
  - independent set formulas
- Prove size lower bounds via technique that doesn't use degree
  - clique formulas
  - weak pigeonhole principle formulas
- Find truly general framework capturing all PC degree lower bounds
  - We generalize only part of [Alekhnovich & Razborov '01]
  - Cannot deal with lower bounds à la [Buss et al. '99]
- Go beyond polynomial calculus (to sums-of-squares, for instance)

Take-away Message

Generalized method for polynomial calculus degree lower bounds

- Unified framework for most previous lower bounds
- Exponential size lower bound for Functional PHP

Future directions

- Extend techniques further to other tricky formulas
- Develop non-degree-based size lower bound techniques

Take-away Message

Generalized method for polynomial calculus degree lower bounds

- Unified framework for most previous lower bounds
- Exponential size lower bound for Functional PHP

Future directions

- Extend techniques further to other tricky formulas
- Develop non-degree-based size lower bound techniques

Thank you for your attention!