I skärningspunkten mellan beviskomplexitet och SAT-lösning

Jakob Nordström

Teorigruppen
KTH Datavetenskap och kommunikation

Docentföreläsning
6 november 2015

Satisfiability (SAT) Problem *[Satisfierbarhetsproblemet]*

$$
\begin{array}{lclc}
(x_1 \lor x_2 \lor \overline{x}_3) & \land & (\overline{x}_1 \lor x_2 \lor \overline{x}_4) & \land \\
(x_1 \lor \overline{x}_3 \lor \overline{x}_4) & \land & (x_2 \lor x_3 \lor \overline{x}_4) & \land \\
(\overline{x}_2 \lor \overline{x}_3 \lor \overline{x}_4) & \land & (\overline{x}_2 \lor x_3 \lor \overline{x}_4) & \land \\
(x_2 \lor \overline{x}_3 \lor x_5) & \land & (\overline{x}_2 \lor x_3 \lor x_5) & \land \\
(\overline{x}_2 \lor x_4 \lor x_5) & \land & (\overline{x}_3 \lor x_4 \lor \overline{x}_5) & \land \\
(x_3 \lor x_4 \lor x_5) & \land & (x_3 \lor x_4 \lor \overline{x}_5) &
\end{array}
$$

Satisfiability (SAT) Problem *[Satisfierbarhetsproblemet]*

$$
\begin{array}{ccccc}
(x_1 \lor x_2 \lor \overline{x}_3) & \land & (\overline{x}_1 \lor x_2 \lor \overline{x}_4) & \land \\
(x_1 \lor \overline{x}_3 \lor \overline{x}_4) & \land & (x_2 \lor x_3 \lor \overline{x}_4) & \land \\
(\overline{x}_2 \lor \overline{x}_3 \lor \overline{x}_4) & \land & (\overline{x}_2 \lor x_3 \lor \overline{x}_4) & \land \\
(x_2 \lor \overline{x}_3 \lor x_5) & \land & (\overline{x}_2 \lor x_3 \lor x_5) & \land \\
(\overline{x}_2 \lor x_4 \lor x_5) & \land & (\overline{x}_3 \lor x_4 \lor \overline{x}_5) & \land \\
(x_3 \lor x_4 \lor x_5) & \land & (x_3 \lor x_4 \lor \overline{x}_5) &
\end{array}
$$

- Variables should be set to true or false

Satisfiability (SAT) Problem *[Satisfierbarhetsproblemet]*

$$
\begin{array}{lllll}
(x_1 \lor x_2 \lor \overline{x}_3) & \land & (\overline{x}_1 \lor x_2 \lor \overline{x}_4) & \land \\
(x_1 \lor \overline{x}_3 \lor \overline{x}_4) & \land & (x_2 \lor x_3 \lor \overline{x}_4) & \land \\
(\overline{x}_2 \lor \overline{x}_3 \lor \overline{x}_4) & \land & (\overline{x}_2 \lor x_3 \lor \overline{x}_4) & \land \\
(x_2 \lor \overline{x}_3 \lor x_5) & \land & (\overline{x}_2 \lor x_3 \lor x_5) & \land \\
(\overline{x}_2 \lor x_4 \lor x_5) & \land & (\overline{x}_3 \lor x_4 \lor \overline{x}_5) & \land \\
(x_3 \lor x_4 \lor x_5) & \land & (x_3 \lor x_4 \lor \overline{x}_5) &
\end{array}
$$

- Variables should be set to true or false
- Constraint $(x_1 \lor x_2 \lor \overline{x}_3)$: means x_1 or x_2 should be true or x_3 false

Satisfiability (SAT) Problem *[Satisfierbarhetsproblemet]*

$$
\begin{array}{lclcl}
(x_1 \lor x_2 \lor \overline{x}_3) & \land & (\overline{x}_1 \lor x_2 \lor \overline{x}_4) & \land \\
(x_1 \lor \overline{x}_3 \lor \overline{x}_4) & \land & (x_2 \lor x_3 \lor \overline{x}_4) & \land \\
(\overline{x}_2 \lor \overline{x}_3 \lor \overline{x}_4) & \land & (\overline{x}_2 \lor x_3 \lor \overline{x}_4) & \land \\
(x_2 \lor \overline{x}_3 \lor x_5) & \land & (\overline{x}_2 \lor x_3 \lor x_5) & \land \\
(\overline{x}_2 \lor x_4 \lor x_5) & \land & (\overline{x}_3 \lor x_4 \lor \overline{x}_5) & \land \\
(x_3 \lor x_4 \lor x_5) & \land & (x_3 \lor x_4 \lor \overline{x}_5) &
\end{array}
$$

- Variables should be set to true or false
- Constraint $(x_1 \lor x_2 \lor \overline{x}_3)$: means x_1 or x_2 should be true or x_3 false
- $\land$ means all constraints should hold simultaneously

Satisfiability (SAT) Problem *[Satisfierbarhetsproblemet]*

$$
\begin{array}{lclcl}
(x_1 \lor x_2 \lor \overline{x}_3) & \land & (\overline{x}_1 \lor x_2 \lor \overline{x}_4) & \land \\
(x_1 \lor \overline{x}_3 \lor \overline{x}_4) & \land & (x_2 \lor x_3 \lor \overline{x}_4) & \land \\
(\overline{x}_2 \lor \overline{x}_3 \lor \overline{x}_4) & \land & (\overline{x}_2 \lor x_3 \lor \overline{x}_4) & \land \\
(x_2 \lor \overline{x}_3 \lor x_5) & \land & (\overline{x}_2 \lor x_3 \lor x_5) & \land \\
(\overline{x}_2 \lor x_4 \lor x_5) & \land & (\overline{x}_3 \lor x_4 \lor \overline{x}_5) & \land \\
(x_3 \lor x_4 \lor x_5) & \land & (x_3 \lor x_4 \lor \overline{x}_5) &
\end{array}
$$

- Variables should be set to true or false
- Constraint $(x_1 \lor x_2 \lor \overline{x}_3)$: means x_1 or x_2 should be true or x_3 false
- $\land$ 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?

Satisfiability (SAT) Problem *[Satisfierbarhetsproblemet]*

$$
\begin{array}{ccccc}
(x_1 \lor x_2 \lor \overline{x}_3) & \land & (\overline{x}_1 \lor x_2 \lor \overline{x}_4) & \land \\
(x_1 \lor \overline{x}_3 \lor \overline{x}_4) & \land & (x_2 \lor x_3 \lor \overline{x}_4) & \land \\
(\overline{x}_2 \lor \overline{x}_3 \lor \overline{x}_4) & \land & (\overline{x}_2 \lor x_3 \lor \overline{x}_4) & \land \\
(x_2 \lor \overline{x}_3 \lor x_5) & \land & (\overline{x}_2 \lor x_3 \lor x_5) & \land \\
(\overline{x}_2 \lor x_4 \lor x_5) & \land & (\overline{x}_3 \lor x_4 \lor \overline{x}_5) & \land \\
(x_3 \lor x_4 \lor x_5) & \land & (x_3 \lor x_4 \lor \overline{x}_5) &
\end{array}
$$

- Variables should be set to true or false
- Constraint $(x_1 \lor x_2 \lor \overline{x}_3)$: means x_1 or x_2 should be true or x_3 false
- $\land$ 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?

Can we use computers to solve the SAT problem efficiently?

Complexity theory

- Propositional satisfiability foundational problem in (Theoretical) Computer Science
- SAT proven NP-complete *[NP-fullständigt]* in 1971
- Hence most likely totally intractable
- Just remains to prove this — one of the million-dollar "Millennium Problems"

Computational Complexity Theory and SAT Solving

Complexity theory

- Propositional satisfiability foundational problem in (Theoretical) Computer Science
- SAT proven NP-complete *[NP-fullständigt]* in 1971
- Hence most likely totally intractable
- Just remains to prove this — one of the million-dollar "Millennium Problems"

Applied SAT solving

- Dramatic performance increase last 15–20 years
- State-of-the-art SAT solvers *[SAT-lösare]* can deal with real-world formulas containing millions of variables
- But best solvers still based on methods from early 1960s
- Also, tiny formulas known that are totally beyond reach

SAT Solving and Proof Complexity

- How can state-of-the-art SAT solvers decide satisfiability of such huge formulas?
- Why do they work so well? And why do they sometimes miserably fail?
- Best current SAT solvers
  - Based on so-called conflict-driven clause learning (CDCL) *[konfliktdriven klausulinlärning]*
  - Sometimes algebraic reasoning (e.g., Gaussian elimination)
  - Sometimes geometric reasoning (e.g., cardinality constraints)
- How can we analyze the power of these methods?
  One of the question addressed by research area of
  proof complexity *[beviskomplexitet]*

Outline of This Presentation

Overview of proof complexity focusing on SAT solving connections:

- Conflict-driven clause learning — resolution *[resolution]*
- Algebraic Gröbner basis computations — polynomial calculus *[polynomkalkyl]*
- Geometric pseudo-Boolean solvers — cutting planes *[skärande plan]*

Survey (some of) what is known about these proof systems

Show theoretical "benchmark formulas" used to understand potential and limitations of methods of reasoning

Some Notation and Terminology

- Literal a: variable x or its negation $\overline{x}$
- Clause $C = a_1 \vee \cdots \vee a_k$: disjunction of literals
  (Consider as sets, so no repetitions and order irrelevant)
- CNF formula $F = C_1 \wedge \cdots \wedge C_m$: conjunction of clauses
- k-CNF formula: CNF formula with clauses of size $\leq k$
  (where k is some constant)
- Mostly assume formulas k-CNFs (for simplicity of exposition)
  Conversion to 3-CNF (most often) doesn't change much
- N denotes size of formula (# literals, which is $\approx$ # clauses)

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 $\perp$ 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

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

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

1.	$x \vee y$	Axiom
2.	$\boldsymbol{x \vee \overline{y} \vee z}$	Axiom
3.	$\overline{x} \vee z$	Axiom
4.	$\boldsymbol{\overline{y} \vee \overline{z}}$	Axiom
5.	$\overline{x} \vee \overline{z}$	Axiom
6.	$x \vee \overline{y}$	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 $\bot$ 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.	$\boldsymbol{x}$	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.	$\boldsymbol{x}$	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.	$\boldsymbol{\overline{x} \lor z}$	Axiom
4.	$\overline{y} \lor \overline{z}$	Axiom
5.	$\boldsymbol{\overline{x} \lor \overline{z}}$	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

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

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.	$\boldsymbol{\overline{x}}$	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)

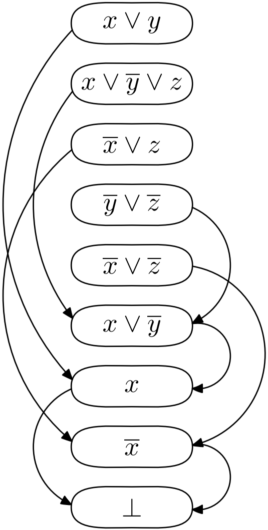
Derive new clauses by resolution rule

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

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

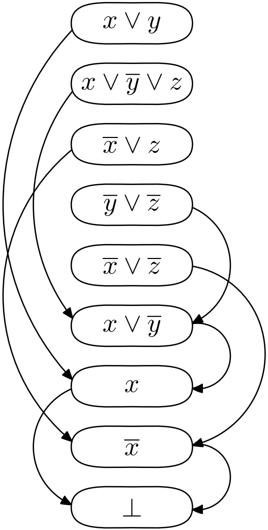
$$\frac{C \lor x \qquad D \lor \overline{x}}{C \lor D}$$

Refutation ends when empty clause $\bot$ derived

Can represent refutation as

- annotated list or
- directed acyclic graph

Tree-like *[trädlik]* resolution if DAG is tree



Resolution Size/Length

Size/length = # clauses in refutation
[Bevisstorlek/-längd]

Most fundamental measure in proof complexity

Lower bound on CDCL running time
(can extract resolution proof from execution trace)

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

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

Examples of Hard Formulas w.r.t Resolution Length (1/3)

Pigeonhole principle (PHP) [Hak85]*
"$n+1$ 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

(*) List of full references given at the end of the slides (available online)

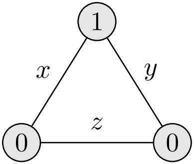
Examples of Hard Formulas w.r.t Resolution Length (2/3)

Tseitin formulas [Urq87]
"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 edges around vertex = label

Requires length $\exp(\Omega(N))$ on well-connected so-called expanders



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

Examples of Hard Formulas w.r.t Resolution Length (3/3)

Random k-CNF formulas [CS88]
Δn randomly sampled k-clauses over n variables

($\Delta \gtrsim 4.5$ sufficient to get unsatisfiable 3-CNF almost surely)

Again lower bound $\exp(\Omega(N))$

Examples of Hard Formulas w.r.t Resolution Length (3/3)

Random k-CNF formulas [CS88]
Δn randomly sampled k-clauses over n variables

($\Delta \gtrsim 4.5$ sufficient to get unsatisfiable 3-CNF almost surely)

Again lower bound $\exp(\Omega(N))$

And more. . .

- k-colourability [BCMM05]
- Independent sets and vertex covers [BIS07]
- Zero-one designs [Spe10, VS10, MN14]
- Et cetera. . .

Resolution Width

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

Resolution Width

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

Width upper bound $\Rightarrow$ length upper bound

Proof: at most $(2 \cdot \#\text{variables})^{\text{width}}$ distinct clauses
(This simple counting argument is essentially tight [ALN14])

Resolution Width

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

Width upper bound $\Rightarrow$ length upper bound

Proof: at most $(2 \cdot \#\text{variables})^{\text{width}}$ distinct clauses
(This simple counting argument is essentially tight [ALN14])

Width lower bound $\Rightarrow$ length lower bound

Much less obvious...

Width Lower Bounds Imply Length Lower Bounds

Theorem ([BW01])

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

Width Lower Bounds Imply Length Lower Bounds

Theorem ([BW01])

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

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

Width Lower Bounds Imply Length Lower Bounds

Theorem ([BW01])

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

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

For tree-like resolution have length $\geq 2^{\text{width}}$ [BW01]

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

Optimality of the Length-Width Lower Bound

Ordering principles [Stå96, BG01]
"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

Can also add "total order" axioms

$x_{i,j} \vee x_{j,i}$	totality; either $e_i < e_j$ or $e_j < e_i$

Refutable in resolution in length $\mathcal{O}(N)$
Requires resolution width $\Omega\left(\sqrt[3]{N}\right)$ (3-CNF version)

Resolution Space

Space = max # clauses in memory when performing refutation *[Bevisminne]*

Motivated by SAT solver memory usage (but also intrinsically interesting for proof complexity)

Can be measured in different ways — focus here on most common measure clause space *[klausulminne]*

Space at step t: # clauses at steps $\leq t$ used at steps $\geq t$

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$(7, 8)$

Resolution Space

Space = max # clauses in memory when performing refutation *[Bevisminne]*

Motivated by SAT solver memory usage (but also intrinsically interesting for proof complexity)

Can be measured in different ways — focus here on most common measure clause space *[klausulminne]*

Space at step t: # clauses at steps $\leq t$ used at steps $\geq t$

Example: Space at step 7 . . .

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$(7, 8)$

Resolution Space

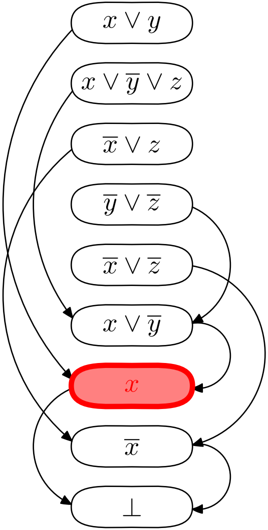
Space = max # clauses in memory when performing refutation *[Bevisminne]*

Motivated by SAT solver memory usage (but also intrinsically interesting for proof complexity)

Can be measured in different ways — focus here on most common measure clause space *[klausulminne]*

Space at step t: # clauses at steps $\leq t$ used at steps $\geq t$

Example: Space at step 7 . . .



Resolution Space

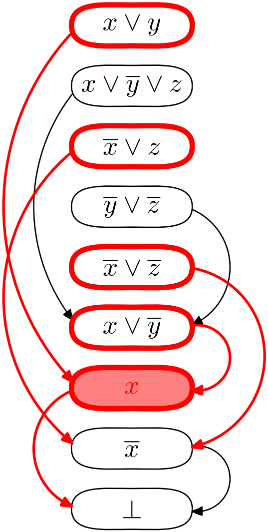
Space = max # clauses in memory when performing refutation *[Bevisminne]*

Motivated by SAT solver memory usage (but also intrinsically interesting for proof complexity)

Can be measured in different ways — focus here on most common measure clause space *[klausulminne]*

Space at step t: # clauses at steps $\leq t$ used at steps $\geq t$

Example: Space at step 7 is 5



Bounds on Resolution Space

Space always at most $N + \mathcal{O}(1)$ (!) [ET01]

Lower bounds for

- Pigeonhole principle [ABRW02, ET01]
- Tseitin formulas [ABRW02, ET01]
- Random k-CNFs [BG03]

Bounds on Resolution Space

Space always at most $N + \mathcal{O}(1)$ (!) [ET01]

Lower bounds for

- Pigeonhole principle [ABRW02, ET01]
- Tseitin formulas [ABRW02, ET01]
- Random k-CNFs [BG03]

Results always matching width bounds

And proofs of very similar flavour. . . What is going on?

Space vs. Width

Theorem ([AD08])

$$\mathit{space} \geq \mathit{width} + \mathcal{O}(1)$$

Space vs. Width

Theorem ([AD08])

$$\mathit{space} \geq \mathit{width} + \mathcal{O}(1)$$

Are space and width asymptotically always the same? **No!**

Space vs. Width

Theorem ([AD08])

$$\textit{space} \geq \textit{width} + \mathcal{O}(1)$$

Are space and width asymptotically always the same? **No!**

Pebbling formulas [BN08]

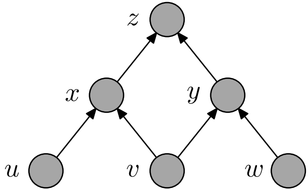
- Can be refuted in width $\mathcal{O}(1)$
- May require space $\Omega(N/\log N)$

A bit more involved to describe than previous benchmarks. . .

Pebbling Formulas: Vanilla Version

CNF formulas encoding so-called pebble games on DAGs

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. $\overline{z}$

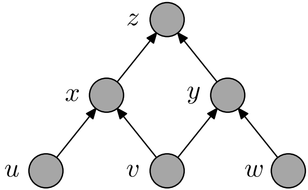


- sources are true
- truth propagates upwards
- but sink is false

Pebbling Formulas: Vanilla Version

CNF formulas encoding so-called pebble games on DAGs

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. $\overline{z}$

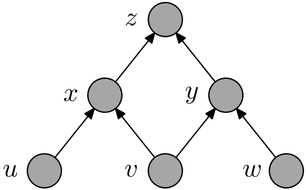


- sources are true
- truth propagates upwards
- but sink is false

Pebbling Formulas: Vanilla Version

CNF formulas encoding so-called pebble games on DAGs

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. $\overline{z}$

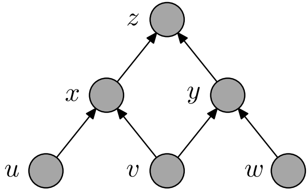


- sources are true
- truth propagates upwards
- but sink is false

Pebbling Formulas: Vanilla Version

CNF formulas encoding so-called pebble games on DAGs

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. $\overline{z}$

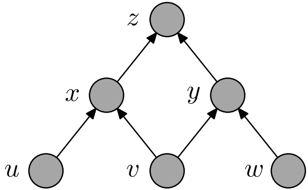


- sources are true
- truth propagates upwards
- but sink is false

Pebbling Formulas: Vanilla Version

CNF formulas encoding so-called pebble games on DAGs

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

Extensive literature on pebbling space and time-space trade-offs from 1970s and 80s

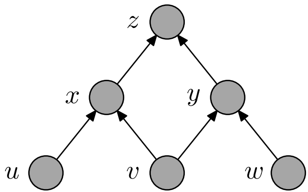
Have been useful in proof complexity before in various contexts

Hope that pebbling properties of DAG somehow carry over to resolution refutations of pebbling formulas.

Pebbling Formulas: Vanilla Version

CNF formulas encoding so-called pebble games on DAGs

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

Extensive literature on pebbling space and time-space trade-offs from 1970s and 80s

Have been useful in proof complexity before in various contexts

Hope that pebbling properties of DAG somehow carry over to resolution refutations of pebbling formulas. **Except. . .**

Substituted Pebbling Formulas

Won't work — pebbling formulas supereasy

Make formula harder by substituting $x_1 \oplus x_2$ for every variable x (also works for other Boolean functions with "right" properties):

$$\overline{x} \vee y$$

$$\Downarrow$$

$$\neg(x_1 \oplus x_2) \vee (y_1 \oplus y_2)$$

$$\Downarrow$$

$$\begin{aligned} & (x_1 \vee \overline{x}_2 \vee y_1 \vee y_2) \\ \wedge\ & (x_1 \vee \overline{x}_2 \vee \overline{y}_1 \vee \overline{y}_2) \\ \wedge\ & (\overline{x}_1 \vee x_2 \vee y_1 \vee y_2) \\ \wedge\ & (\overline{x}_1 \vee x_2 \vee \overline{y}_1 \vee \overline{y}_2) \end{aligned}$$

Now CNF formula inherits pebbling graph properties!

Space-Width Trade-offs

Given a formula easy w.r.t. these complexity measures, can refutations be optimized for two or more measures?

For space vs. width, the answer is a strong no

Theorem ([Ben09])

There are formulas for which

- *exist refutations in width $\mathcal{O}(1)$*
- *exist refutations in space $\mathcal{O}(1)$*
- *optimization of one measure causes (essentially) worst-case behaviour for other measure*

Holds for vanilla version of pebbling formulas

Length-Space Trade-offs

Theorem ([BN11, BBI12, BNT13])

There are formulas for which

- *exist refutations in short length*
- *exist refutations in small space*
- *optimization of one measure* causes *dramatic blow-up for other measure*

Holds for

- Substituted pebbling formulas over the right graphs
- Tseitin formulas over long, narrow rectangular grids

So no meaningful simultaneous optimization possible for length and space in the worst case

Length-Width Trade-offs

What about length versus width? [BW01] transforms short refutation to narrow one, but blows up length exponentially

- Is this blow-up inherent?
- Or just an artifact of the proof?

Length-Width Trade-offs

What about length versus width? [BW01] transforms short refutation to narrow one, but blows up length exponentially

- Is this blow-up inherent?
- Or just an artifact of the proof?

Recent news (solved after problem was advertised at *SAT '14*):

Theorem ([Tha14])

There are formulas for which

- *exist refutations in short length*
- *exist refutations in small width*
- *optimization of one measure* causes *dramatic blow-up for other measure*

Minor issue: formulas have logarithmic width — would like k-CNFs

Recap of Complexity Measures for Resolution

Recall that N = size of formula

Length

\# clauses in refutation — at most $\exp(N)$

Width

Size of largest clause in refutation — at most N

Space

Max # clauses one needs to remember when "verifying correctness of refutation" — at most N (!)

Proof Complexity Measures and CDCL Hardness

Recall $\log(\text{length}) \lesssim$ width $\lesssim$ space

Proof Complexity Measures and CDCL Hardness

Recall $\log(\text{length}) \lesssim$ width $\lesssim$ space

Length

- Lower bound on running time for CDCL
- CDCL polynomially simulates resolution [PD11]
- But short proofs may be worst-case intractable to find [AR08]

Proof Complexity Measures and CDCL Hardness

Recall $\log(\text{length}) \lesssim \text{width} \lesssim \text{space}$

Length

- Lower bound on running time for CDCL
- CDCL polynomially simulates resolution [PD11]
- But short proofs may be worst-case intractable to find [AR08]

Width

- Searching in small width known heuristic in AI community
- Small width $\Rightarrow$ CDCL solver will run fast [AFT11]

Proof Complexity Measures and CDCL Hardness

Recall $\log(\text{length}) \lesssim$ width $\lesssim$ space

Length

- Lower bound on running time for CDCL
- CDCL polynomially simulates resolution [PD11]
- But short proofs may be worst-case intractable to find [AR08]

Width

- Searching in small width known heuristic in AI community
- Small width $\Rightarrow$ CDCL solver will run fast [AFT11]

Space

- In practice, memory consumption important bottleneck
- Space complexity gives lower bound on clause database size
- Plus assumes solver knows **exactly** which clauses to keep $\Rightarrow$ in reality, probably (much) more memory needed

Relations Between Theoretical and Practical Hardness?

1. Are width or even space lower bounds relevant indicators of CDCL hardness?
2. Or is it true in practice that CDCL does essentially as well as resolution w.r.t. length/running time? (Although only small part of resolution search space explored)
3. Can CDCL even do as well as resolution w.r.t. time and space simultaneously?

Relations Between Theoretical and Practical Hardness?

1. Are width or even space lower bounds relevant indicators of CDCL hardness?
2. Or is it true in practice that CDCL does essentially as well as resolution w.r.t. length/running time? (Although only small part of resolution search space explored)
3. Can CDCL even do as well as resolution w.r.t. time and space simultaneously?

Not mathematically well-defined questions...

But perhaps still possible to perform experiments and draw interesting conclusions?

Practical Conclusions So Far?

- Some preliminary work along these lines in [JMNŽ12]
- But messy reality not easily captured by nice theories
- CDCL performance on combinatorial benchmarks sometimes surprising; e.g.:
  - For PHP, worse behaviour with heuristics than without
  - Sometimes "easy" formulas harder than "hard" ones?! [MN14]
  - Sometimes minor changes in internal parameters makes all the difference between supereasy and totally impossible

Open Problems

- *Could explanations of above phenomena help us understand CDCL better?*
- *Could controlled experiments on easily scalable theoretical benchmarks yield other interesting insights?*

Polynomial Calculus *[Polynomkalkyl]*

Introduced in [CEI96]; below modified version from [ABRW02]

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 $0 \equiv \mathit{true}$ and $1 \equiv \mathit{false}$)

Polynomial Calculus *[Polynomkalkyl]*

Introduced in [CEI96]; below modified version from [ABRW02]

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

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

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

Size, Degree and Space

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

Size, Degree and Space

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

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

(Can also define length measure — might be much smaller since polynomials can be of exponential size)

Degree — analogue of resolution width
largest degree of monomial in refutation
[bevisgrad]

(Monomial) space — analogue of resolution (clause) space
max # monomials in memory during refutation (with repetitions)
[bevismonomminne]

Polynomial Calculus Strictly Stronger than Resolution

Polynomial calculus simulates resolution efficiently with respect to length/size, width/degree, and space simultaneously

- Can mimic resolution refutation step by step
- Hence worst-case upper bounds for resolution carry over

Polynomial Calculus Strictly Stronger than Resolution

Polynomial calculus simulates resolution efficiently with respect to length/size, width/degree, and space simultaneously

- Can mimic resolution refutation step by step
- Hence worst-case upper bounds for resolution carry over

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

- Tseitin formulas on expanders (just do Gaussian elimination)
- Onto functional pigeonhole principle [Rii93]

Polynomial Calculus Strictly Stronger than Resolution

Polynomial calculus simulates resolution efficiently with respect to length/size, width/degree, and space simultaneously

- Can mimic resolution refutation step by step
- Hence worst-case upper bounds for resolution carry over

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

- Tseitin formulas on expanders (just do Gaussian elimination)
- Onto functional pigeonhole principle [Rii93]

Open Problem

Decide whether polynomial calculus is strictly stronger than resolution w.r.t. space

Size vs. Degree

- Degree upper bound ⇒ size upper bound [CEI96]
  Qualitatively similar to resolution bound
  A bit more involved argument
  Again essentially tight by [ALN14]

- Degree lower bound ⇒ size lower bound [IPS99]
  Precursor of [BW01] — can do same proof to get same bound

- Size-degree lower bound essentially optimal [GL10]
  Example: same ordering principle formulas

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

Examples of Hard Formulas w.r.t. Size (and Degree)

Pigeonhole principle formulas

Follows from [AR03]

Earlier work on other encodings in [Raz98, IPS99]

Hard even with functionality axioms added [MN15]

Tseitin formulas with "wrong modulus"

Can define Tseitin-like formulas counting $\mathrm{mod}\ p$ for $p \neq 2$

Hard if $p \neq$ characteristic of field [BGIP01]

Random k-CNF formulas

Hard in all characteristics except 2 [BI99]

Lower bound for all characteristics in [AR03]

Bounds on Polynomial Calculus Space

Lower bound for PHP with wide clauses [ABRW02]

k-CNF formulas much trickier — sequence of lower bounds for

- Obfuscated 4-CNF versions of PHP [FLN+15]
- Random 4-CNFs [BG15]
- Tseitin formulas in 4-CNF on (some) expanders [FLM+13] (but results not tight)
- Random 3-CNFs [BBG+15]

Open Problems

Prove polynomial calculus space lower bounds on

- *Tseitin formulas on any expander*
- *3-CNF version of PHP formulas*

Space vs. Degree

Open Problem (analogue of [AD08])

Is it true that space $\geq$ degree $+ \mathcal{O}(1)$?

Some partial results in [FLM+13] (but weak)

Space vs. Degree

Open Problem (analogue of [AD08])

Is it true that space $\geq$ degree $+ \mathcal{O}(1)$?

Some partial results in [FLM$^+$13] (but weak)

Optimal separation of space and degree in [FLM$^+$13] using flavour of Tseitin formulas which

- can be refuted in degree $\mathcal{O}(1)$
- require space $\Omega(N)$
- but separating formulas depend on characteristic of field

Open Problem

Prove space lower bounds for substituted pebbling formulas (would give space-degree separation independent of characteristic)

Trade-offs for Polynomial Calculus

- Strong, essentially optimal space-degree trade-offs [BNT13]
  Same vanilla pebbling formulas as for resolution
  Same parameters

- Strong size-space trade-offs [BNT13]
  Same formulas as for resolution
  Some loss in parameters

Open Problem

Are there size-degree trade-offs in polynomial calculus?

[Tha14] works only for resolution (so far)

Algebraic SAT Solvers?

- Quite some excitement about Gröbner basis approach to SAT solving after [CEI96]
- Promise of performance improvement failed to deliver
- Meanwhile: the CDCL revolution. . .

Algebraic SAT Solvers?

- Quite some excitement about Gröbner basis approach to SAT solving after [CEI96]
- Promise of performance improvement failed to deliver
- Meanwhile: the CDCL revolution...
- Some current SAT solvers do Gaussian elimination, but this is only very limited form of polynomial calculus
- Is it harder to build good algebraic SAT solvers, or is it just that too little work has been done (or both)?
- Some shortcut seems to be needed — full Gröbner basis computation does too much work (counts #satisfying assignments — we just want to know whether $\neq 0$)

Cutting Planes *[Skärande plan]*

Introduced in [CCT87]

Clauses interpreted as linear inequalities over the reals with integer coefficients
Example: $x \lor y \lor \overline{z}$ gets translated to $x + y + (1 - z) \geq 1$
(Now $1 \equiv \mathit{true}$ and $0 \equiv \mathit{false}$ again)

Cutting Planes *[Skärande plan]*

Introduced in [CCT87]

Clauses interpreted as linear inequalities over the reals with integer coefficients

Example: $x \vee y \vee \overline{z}$ gets translated to $x + y + (1 - z) \geq 1$

(Now $1 \equiv \mathit{true}$ and $0 \equiv \mathit{false}$ again)

Derivation rules

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

Goal: Derive $0 \geq 1 \Leftrightarrow$ formula unsatisfiable

Size, Length and Space

Length = total # lines/inequalities in refutation

Size = sum also size of coefficients

Space = max # lines in memory during refutation

No (useful) analogue of width/degree

Size, Length and Space

Length = total # lines/inequalities in refutation

Size = sum also size of coefficients

Space = max # lines in memory during refutation

No (useful) analogue of width/degree

Cutting planes

- simulates resolution efficiently w.r.t. length/size and space simultaneously

Size, Length and Space

Length = total # lines/inequalities in refutation

Size = sum also size of coefficients

Space = max # lines in memory during refutation

No (useful) analogue of width/degree

Cutting planes

- simulates resolution efficiently w.r.t. length/size and space simultaneously
- is strictly stronger w.r.t. length/size — can refute PHP efficiently [CCT87]

Size, Length and Space

Length = total # lines/inequalities in refutation

Size = sum also size of coefficients

Space = max # lines in memory during refutation

No (useful) analogue of width/degree

Cutting planes

- simulates resolution efficiently w.r.t. length/size and space simultaneously
- is strictly stronger w.r.t. length/size — can refute PHP efficiently [CCT87]
- is strictly stronger w.r.t. space — can refute any CNF in constant space 5 (!) [GPT15]

Size, Length and Space

Length = total # lines/inequalities in refutation

Size = sum also size of coefficients

Space = max # lines in memory during refutation

No (useful) analogue of width/degree

Cutting planes

- simulates resolution efficiently w.r.t. length/size and space simultaneously
- is strictly stronger w.r.t. length/size — can refute PHP efficiently [CCT87]
- is strictly stronger w.r.t. space — can refute any CNF in constant space 5 (!) [GPT15] (But coefficients will be exponentially large — what if also coefficient size counted?)

Hard Formulas w.r.t Cutting Planes Length

Clique-coclique formulas [Pud97]
"A graph with a k-clique is not $(k-1)$-colourable"

Lower bound via interpolation and circuit complexity

Hard Formulas w.r.t Cutting Planes Length

Clique-coclique formulas [Pud97]
"A graph with a k-clique is not $(k-1)$-colourable"

Lower bound via interpolation and circuit complexity

Open Problems

Prove length lower bounds for cutting planes

- *for Tseitin formulas*
- *for random k-CNFs*
- *for any formula using other technique than interpolation*

Size-Space Trade-offs for Cutting Planes?

- Short cutting planes refutations of Tseitin formulas on expanders require large space [GP14]
  (But such short refutations probably don't exist anyway)
- Short cutting planes refutations of (some) pebbling formulas require large space [HN12, GP14] (such refutations exist)

Size-Space Trade-offs for Cutting Planes?

- Short cutting planes refutations of Tseitin formulas on expanders require large space [GP14]
  (But such short refutations probably don't exist anyway)
- Short cutting planes refutations of (some) pebbling formulas require large space [HN12, GP14] (such refutations exist)

Open Problems

- *Are there trade-offs where the space-efficient CP refutations have small coefficients? (Say, of polynomial size)*
- *Are there space lower bounds for CP refutations with polynomial-size coefficients?*

Already coefficients of absolute size ≤ 2 quite powerful — can refute PHP formulas efficiently [GPT15]

Geometric SAT Solvers?

- There are some so-called pseudo-Boolean SAT solvers *[pseudo-Booleska SAT-lösare]* using (subset of) cutting planes reasoning
- Seems hard to make competitive with CDCL

Geometric SAT Solvers?

- There are some so-called pseudo-Boolean SAT solvers *[pseudo-Booleska SAT-lösare]* using (subset of) cutting planes reasoning
- Seems hard to make competitive with CDCL
- **Roadblock 1:** Given CNF input, solvers cannot discover and use cardinality constraints (too limited form of addition)
- But given helpful encoding, solvers can do really well (e.g., PHP formulas and zero-one designs) [BBLM14]

Geometric SAT Solvers?

- There are some so-called pseudo-Boolean SAT solvers *[pseudo-Booleska SAT-lösare]* using (subset of) cutting planes reasoning
- Seems hard to make competitive with CDCL
- **Roadblock 1:** Given CNF input, solvers cannot discover and use cardinality constraints (too limited form of addition)
- But given helpful encoding, solvers can do really well (e.g., PHP formulas and zero-one designs) [BBLM14]
- **Roadblock 2(?):** Solvers seem inefficient for systems of inequalities that have rational but not integral solutions (too limited form of division?)
- Not well understood at all — work in progress

Summing up This Presentation

Overview of resolution, polynomial calculus and cutting planes
(More details in survey paper [Nor15])

- Resolution fairly well understood
- Polynomial calculus less so
- Cutting planes almost not at all

Summing up This Presentation

Overview of resolution, polynomial calculus and cutting planes
(More details in survey paper [Nor15])

- Resolution fairly well understood
- Polynomial calculus less so
- Cutting planes almost not at all

Open problems motivated by applied SAT solving

- Can proof complexity measures shed more light on the hardness (or easiness) of SAT?
- Is it possible to build efficient SAT solvers based on stronger proof systems than resolution?

Summing up This Presentation

Overview of resolution, polynomial calculus and cutting planes
(More details in survey paper [Nor15])

- Resolution fairly well understood
- Polynomial calculus less so
- Cutting planes almost not at all

Open problems motivated by applied SAT solving

- Can proof complexity measures shed more light on the hardness (or easiness) of SAT?
- Is it possible to build efficient SAT solvers based on stronger proof systems than resolution?

Thank you for your attention!

References I

[ABRW02] Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Space complexity in propositional calculus. *SIAM Journal on Computing*, 31(4):1184–1211, 2002. Preliminary version in *STOC '00*.

[AD08] Albert Atserias and Víctor Dalmau. A combinatorial characterization of resolution width. *Journal of Computer and System Sciences*, 74(3):323–334, May 2008. Preliminary version in *CCC '03*.

[AFT11] Albert Atserias, Johannes Klaus Fichte, and Marc Thurley. Clause-learning algorithms with many restarts and bounded-width resolution. *Journal of Artificial Intelligence Research*, 40:353–373, January 2011. Preliminary version in *SAT '09*.

[ALN14] Albert Atserias, Massimo Lauria, and Jakob Nordström. Narrow proofs may be maximally long. In *Proceedings of the 29th Annual IEEE Conference on Computational Complexity (CCC '14)*, pages 286–297, June 2014.

References II

[AR03] Michael Alekhnovich and Alexander A. Razborov. Lower bounds for polynomial calculus: Non-binomial case. *Proceedings of the Steklov Institute of Mathematics*, 242:18–35, 2003. Available at `http://people.cs.uchicago.edu/~razborov/files/misha.pdf`. Preliminary version in *FOCS '01*.

[AR08] Michael Alekhnovich and Alexander A. Razborov. Resolution is not automatizable unless W[P] is tractable. *SIAM Journal on Computing*, 38(4):1347–1363, October 2008. Preliminary version in *FOCS '01*.

[BBG+15] Patrick Bennett, Ilario Bonacina, Nicola Galesi, Tony Huynh, Mike Molloy, and Paul Wollan. Space proof complexity for random 3-CNFs. Technical Report 1503.01613, arXiv.org, April 2015.

[BBI12] Paul Beame, Chris Beck, and Russell Impagliazzo. Time-space tradeoffs in resolution: Superpolynomial lower bounds for superlinear space. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC '12)*, pages 213–232, May 2012.

References III

[BBLM14] Armin Biere, Daniel Le Berre, Emmanuel Lonca, and Norbert Manthey. Detecting cardinality constraints in CNF. In *Proceedings of the 17th International Conference on Theory and Applications of Satisfiability Testing (SAT '14)*, volume 8561 of *Lecture Notes in Computer Science*, pages 285–301. Springer, July 2014.

[BCMM05] Paul Beame, Joseph C. Culberson, David G. Mitchell, and Cristopher Moore. The resolution complexity of random graph k-colorability. *Discrete Applied Mathematics*, 153(1-3):25–47, December 2005.

[Ben09] Eli Ben-Sasson. Size-space tradeoffs for resolution. *SIAM Journal on Computing*, 38(6):2511–2525, May 2009. Preliminary version in *STOC '02*.

[BG01] María Luisa Bonet and Nicola Galesi. Optimality of size-width tradeoffs for resolution. *Computational Complexity*, 10(4):261–276, December 2001. Preliminary version in *FOCS '99*.

[BG03] Eli Ben-Sasson and Nicola Galesi. Space complexity of random formulae in resolution. *Random Structures and Algorithms*, 23(1):92–109, August 2003. Preliminary version in *CCC '01*.

References IV

[BG15] Ilario Bonacina and Nicola Galesi. A framework for space complexity in algebraic proof systems. *Journal of the ACM*, 62(3):23:1–23:20, June 2015. Preliminary version in *ITCS '13*.

[BGIP01] Samuel R. Buss, Dima Grigoriev, Russell Impagliazzo, and Toniann Pitassi. Linear gaps between degrees for the polynomial calculus modulo distinct primes. *Journal of Computer and System Sciences*, 62(2):267–289, March 2001. Preliminary version in *CCC '99*.

[BI99] Eli Ben-Sasson and Russell Impagliazzo. Random CNF's are hard for the polynomial calculus. In *Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS '99)*, pages 415–421, October 1999. Journal version in [BI10].

[BI10] Eli Ben-Sasson and Russell Impagliazzo. Random CNF's are hard for the polynomial calculus. *Computational Complexity*, 19:501–519, 2010. Preliminary version in *FOCS '99*.

[BIS07] Paul Beame, Russell Impagliazzo, and Ashish Sabharwal. The resolution complexity of independent sets and vertex covers in random graphs. *Computational Complexity*, 16(3):245–297, October 2007.

References V

[BN08] Eli Ben-Sasson and Jakob Nordström. Short proofs may be spacious: An optimal separation of space and length in resolution. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS '08)*, pages 709–718, October 2008.

[BN11] Eli Ben-Sasson and Jakob Nordström. Understanding space in proof complexity: Separations and trade-offs via substitutions. In *Proceedings of the 2nd Symposium on Innovations in Computer Science (ICS '11)*, pages 401–416, January 2011.

[BNT13] Chris Beck, Jakob Nordström, and Bangsheng Tang. Some trade-off results for polynomial calculus. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC '13)*, pages 813–822, May 2013.

[BW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. *Journal of the ACM*, 48(2):149–169, March 2001. Preliminary version in *STOC '99*.

References VI

[CCT87] William Cook, Collette Rene Coullard, and György Turán. On the complexity of cutting-plane proofs. *Discrete Applied Mathematics*, 18(1):25–38, November 1987.

[CEI96] Matthew Clegg, Jeffery Edmonds, and Russell Impagliazzo. Using the Groebner basis algorithm to find proofs of unsatisfiability. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC '96)*, pages 174–183, May 1996.

[CS88] Vašek Chvátal and Endre Szemerédi. Many hard examples for resolution. *Journal of the ACM*, 35(4):759–768, October 1988.

[ET01] Juan Luis Esteban and Jacobo Torán. Space bounds for resolution. *Information and Computation*, 171(1):84–97, 2001. Preliminary versions of these results in *STACS '99* and *CSL '99*.

References VII

[FLM+13] Yuval Filmus, Massimo Lauria, Mladen Mikša, Jakob Nordström, and Marc Vinyals. Towards an understanding of polynomial calculus: New separations and lower bounds (extended abstract). In *Proceedings of the 40th International Colloquium on Automata, Languages and Programming (ICALP '13)*, volume 7965 of *Lecture Notes in Computer Science*, pages 437–448. Springer, July 2013.

[FLN+15] Yuval Filmus, Massimo Lauria, Jakob Nordström, Noga Ron-Zewi, and Neil Thapen. Space complexity in polynomial calculus. *SIAM Journal on Computing*, 44(4):1119–1153, August 2015. Preliminary version in *CCC '12*.

[GL10] Nicola Galesi and Massimo Lauria. Optimality of size-degree trade-offs for polynomial calculus. *ACM Transactions on Computational Logic*, 12:4:1–4:22, November 2010.

[GP14] Mika Göös and Toniann Pitassi. Communication lower bounds via critical block sensitivity. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC '14)*, pages 847–856, May 2014.

References VIII

[GPT15] Nicola Galesi, Pavel Pudlák, and Neil Thapen. The space complexity of cutting planes refutations. In *Proceedings of the 30th Annual Computational Complexity Conference (CCC '15)*, volume 33 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 433–447, June 2015.

[Hak85] Armin Haken. The intractability of resolution. *Theoretical Computer Science*, 39(2-3):297–308, August 1985.

[HN12] Trinh Huynh and Jakob Nordström. On the virtue of succinct proofs: Amplifying communication complexity hardness to time-space trade-offs in proof complexity (extended abstract). In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC '12)*, pages 233–248, May 2012.

[IPS99] Russell Impagliazzo, Pavel Pudlák, and Jiří Sgall. Lower bounds for the polynomial calculus and the Gröbner basis algorithm. *Computational Complexity*, 8(2):127–144, 1999.

References IX

[JMNŽ12] Matti Järvisalo, Arie Matsliah, Jakob Nordström, and Stanislav Živný. Relating proof complexity measures and practical hardness of SAT. In *Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming (CP '12)*, volume 7514 of *Lecture Notes in Computer Science*, pages 316–331. Springer, October 2012.

[MN14] Mladen Mikša and Jakob Nordström. Long proofs of (seemingly) simple formulas. In *Proceedings of the 17th International Conference on Theory and Applications of Satisfiability Testing (SAT '14)*, volume 8561 of *Lecture Notes in Computer Science*, pages 121–137. Springer, July 2014.

[MN15] Mladen Mikša and Jakob Nordström. A generalized method for proving polynomial calculus degree lower bounds. In *Proceedings of the 30th Annual Computational Complexity Conference (CCC '15)*, volume 33 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 467–487, June 2015.

[Nor15] Jakob Nordström. On the interplay between proof complexity and SAT solving. *ACM SIGLOG News*, 2(3):19–44, July 2015.

References X

[PD11] Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artificial Intelligence*, 175:512–525, February 2011. Preliminary version in *CP '09*.

[Pud97] Pavel Pudlák. Lower bounds for resolution and cutting plane proofs and monotone computations. *Journal of Symbolic Logic*, 62(3):981–998, September 1997.

[Raz98] Alexander A. Razborov. Lower bounds for the polynomial calculus. *Computational Complexity*, 7(4):291–324, December 1998.

[Rii93] Søren Riis. *Independence in Bounded Arithmetic*. PhD thesis, University of Oxford, 1993.

[Spe10] Ivor Spence. sgen1: A generator of small but difficult satisfiability benchmarks. *Journal of Experimental Algorithmics*, 15:1.2:1.1–1.2:1.15, March 2010.

[Stå96] Gunnar Stålmarck. Short resolution proofs for a sequence of tricky formulas. *Acta Informatica*, 33(3):277–280, May 1996.

References XI

[Tha14] Neil Thapen. A trade-off between length and width in resolution. Technical Report TR14-137, Electronic Colloquium on Computational Complexity (ECCC), October 2014.

[Urq87] Alasdair Urquhart. Hard examples for resolution. *Journal of the ACM*, 34(1):209–219, January 1987.

[VS10] Allen Van Gelder and Ivor Spence. Zero-one designs produce small hard SAT instances. In *Proceedings of the 13th International Conference on Theory and Applications of Satisfiability Testing (SAT '10)*, volume 6175 of *Lecture Notes in Computer Science*, pages 388–397. Springer, July 2010.