Understanding Conflict-Driven SAT Solving Through the Lens of Proof Complexity

Jakob Nordström

Theoretical Computer Science Group
KTH Royal Institute of Technology

Theory reading group
November 20, 2017

Understanding Conflict-Driven SAT Solving Through the Lens of Proof Complexity?

Jakob Nordström

Theoretical Computer Science Group
KTH Royal Institute of Technology

Theory reading group
November 20, 2017

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)

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

- 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

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

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

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?

SAT Solving in Theory and Practice

The unreasonable effectiveness of SAT solvers

- The Boolean satisfiability problem (SAT) is NP-complete and so should be exponentially hard
- Yet conflict-driven clause learning (CDCL) SAT solvers can deal with formulas containing millions of variables
- How can they work so well? What are their limits?

SAT Solving in Theory and Practice

The unreasonable effectiveness of SAT solvers

- The Boolean satisfiability problem (SAT) is NP-complete and so should be exponentially hard
- Yet conflict-driven clause learning (CDCL) SAT solvers can deal with formulas containing millions of variables
- How can they work so well? What are their limits?

How to understand the power of CDCL?

SAT Solving in Theory and Practice

The unreasonable effectiveness of SAT solvers

- The Boolean satisfiability problem (SAT) is NP-complete and so should be exponentially hard
- Yet conflict-driven clause learning (CDCL) SAT solvers can deal with formulas containing millions of variables
- How can they work so well? What are their limits?

How to understand the power of CDCL?

- Community structure

SAT Solving in Theory and Practice

The unreasonable effectiveness of SAT solvers

- The Boolean satisfiability problem (SAT) is NP-complete and so should be exponentially hard
- Yet conflict-driven clause learning (CDCL) SAT solvers can deal with formulas containing millions of variables
- How can they work so well? What are their limits?

How to understand the power of CDCL?

- Community structure
- Parameterized complexity

SAT Solving in Theory and Practice

The unreasonable effectiveness of SAT solvers

- The Boolean satisfiability problem (SAT) is NP-complete and so should be exponentially hard
- Yet conflict-driven clause learning (CDCL) SAT solvers can deal with formulas containing millions of variables
- How can they work so well? What are their limits?

How to understand the power of CDCL?

- Community structure
- Parameterized complexity
- This talk: **proof complexity**
  Rigorous analysis of underlying method of reasoning

Purpose of This Presentation

- Survey some of the research in the area (most of it **not** mine) including some ongoing work (of mine)
- Discuss some theoretical "benchmark formulas" used to understand potential and limitations of SAT solvers
- Highlight some (of the many) remaining open problems

Purpose of This Presentation

- Survey some of the research in the area (most of it **not** mine) including some ongoing work (of mine)
- Discuss some theoretical "benchmark formulas" used to understand potential and limitations of SAT solvers
- Highlight some (of the many) remaining open problems

Caveats:

- By necessity, selective and somewhat subjective coverage
- Will sweep some technical details under the rug — happy to discuss offline
- Full references for all papers at end of slides

Some More Caveats and Clarifications

Limitations of proof complexity

- Asking for rigorous analysis is asking a lot. . .
- In addition, proof complexity considers optimal algorithms (so restrict focus to unsatisfiable formulas)
- Still possible to prove some highly nontrivial theorems
- Separate question how to interpret these theoretical theorems

Some More Caveats and Clarifications

Limitations of proof complexity

- Asking for rigorous analysis is asking a lot...
- In addition, proof complexity considers optimal algorithms (so restrict focus to unsatisfiable formulas)
- Still possible to prove some highly nontrivial theorems
- Separate question how to interpret these theoretical theorems

Why focus on theory benchmarks?

- See what SAT solvers can do (sometimes very neat things)
- See what SAT solvers cannot do (provably hard instances)
- See what SAT solvers "should be able" to do (formulas easy for proof system but hard for corresponding SAT solvers)

Outline

Some Notation and Terminology

- Literal a: variable x or its negation $\overline{x}$ (or $\neg x$)
- 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 denotes size of formula (# literals counted with repetitions)
- $\mathcal{O}(f(N))$ grows at most as quickly as $f(N)$ asymptotically
  $\Omega(g(N))$ grows at least as quickly as $g(N)$ asymptotically
  $\Theta(h(N))$ grows equally quickly as $h(N)$ asymptotically

The Resolution Proof System Underlying CDCL

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Done when empty clause $\bot$ derived

The Resolution Proof System Underlying CDCL

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Done when empty clause $\perp$ 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 Underlying CDCL

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Done when empty clause $\perp$ derived

Can represent refutation/proof 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.	x	Res$(1, 6)$
8.	$\overline{x}$	Res$(3, 5)$
9.	$\perp$	Res$(7, 8)$

The Resolution Proof System Underlying CDCL

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Done when empty clause $\perp$ derived

Can represent refutation/proof 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.	$\perp$	Res$(7, 8)$

The Resolution Proof System Underlying CDCL

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Done when empty clause $\bot$ derived

Can represent refutation/proof 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 Underlying CDCL

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Done when empty clause $\bot$ derived

Can represent refutation/proof 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 Underlying CDCL

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Done when empty clause $\bot$ derived

Can represent refutation/proof as

- annotated list or
- directed acyclic graph

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

The Resolution Proof System Underlying CDCL

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Done when empty clause $\bot$ derived

Can represent refutation/proof 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.	x	Res$(1,6)$
8.	$\overline{x}$	Res$(3,5)$
9.	$\bot$	Res$(7,8)$

The Resolution Proof System Underlying CDCL

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Done when empty clause $\perp$ derived

Can represent refutation/proof 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.	$\perp$	Res$(7,8)$

The Resolution Proof System Underlying CDCL

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Done when empty clause $\bot$ derived

Can represent refutation/proof 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 Underlying CDCL

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Done when empty clause $\bot$ derived

Can represent refutation/proof 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.	$\boldsymbol{\overline{x}}$	Res$(3,5)$
9.	$\bot$	Res$(7,8)$

The Resolution Proof System Underlying CDCL

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Done when empty clause $\perp$ derived

Can represent refutation/proof 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.	x	Res$(1,6)$
8.	$\boldsymbol{\overline{x}}$	Res$(3,5)$
9.	$\perp$	Res$(7,8)$

The Resolution Proof System Underlying CDCL

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Done when empty clause $\perp$ derived

Can represent refutation/proof 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.	$\perp$	Res$(7,8)$

The Resolution Proof System Underlying CDCL

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Done when empty clause $\perp$ derived

Can represent refutation/proof 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.	$\perp$	Res$(7,8)$

The Resolution Proof System Underlying CDCL

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Done when empty clause $\bot$ derived

Can represent refutation/proof 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 Underlying CDCL

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

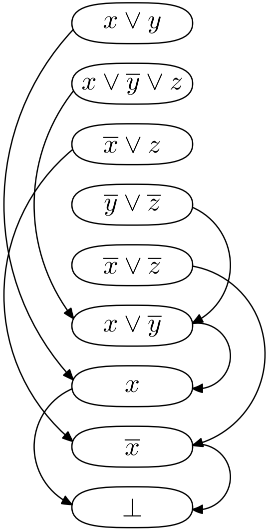
Derive new clauses by resolution rule

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

Done when empty clause $\bot$ derived

Can represent refutation/proof as

- annotated list or
- directed acyclic graph



The Resolution Proof System Underlying CDCL

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

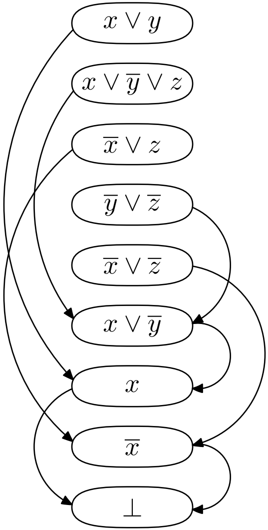
$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D}$$

Done when empty clause $\perp$ derived

Can represent refutation/proof as

- annotated list or
- directed acyclic graph

Tree-like resolution if DAG is tree



Making the Connection to DPLL

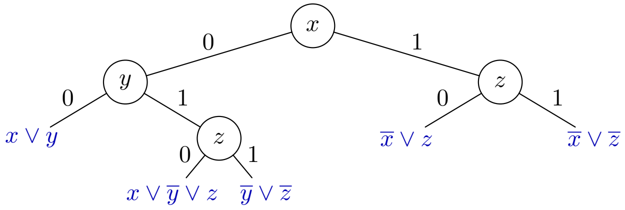
Basis of best modern SAT solvers still DPLL method [DP60, DLL62]

Making the Connection to DPLL

Basis of best modern SAT solvers still DPLL method [DP60, DLL62]

Visualize execution of DPLL algorithm as search tree

- Branch on variable assignments in internal nodes
- Stop in leaves when falsfied clause found



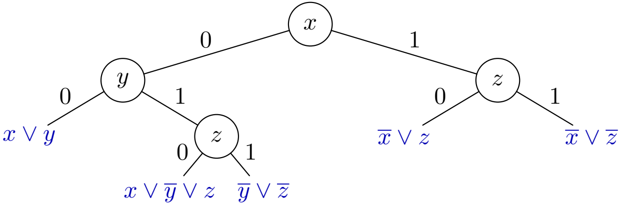
DPLL Execution as Resolution Proof

A DPLL execution is a resolution proof

DPLL Execution as Resolution Proof

A DPLL execution is a resolution proof

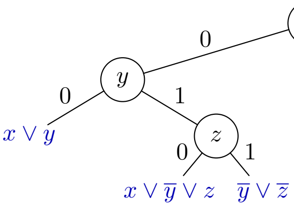
Look at our example again:



DPLL Execution as Resolution Proof

A DPLL execution is a resolution proof

Look at our example again:

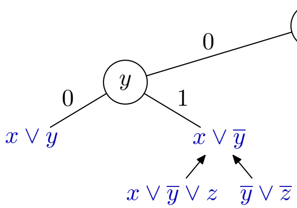


and apply resolution rule bottom-up

DPLL Execution as Resolution Proof

A DPLL execution is a resolution proof

Look at our example again:

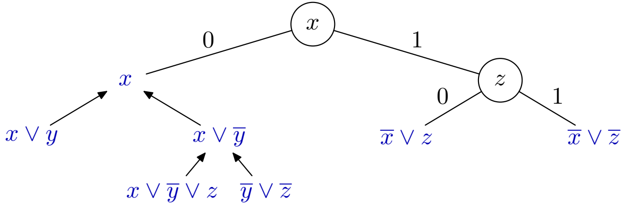


and apply resolution rule bottom-up

DPLL Execution as Resolution Proof

A DPLL execution is a resolution proof

Look at our example again:

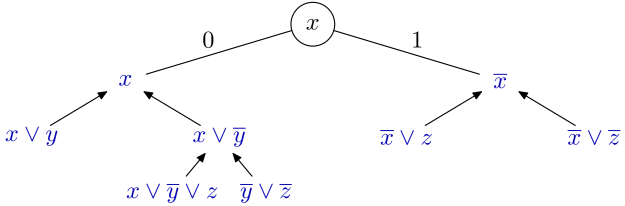


and apply resolution rule bottom-up

DPLL Execution as Resolution Proof

A DPLL execution is a resolution proof

Look at our example again:

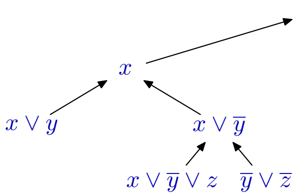


and apply resolution rule bottom-up

DPLL Execution as Resolution Proof

A DPLL execution is a resolution proof

Look at our example again:

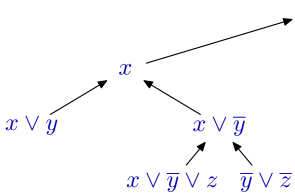


and apply resolution rule bottom-up

DPLL Execution as Resolution Proof

A DPLL execution is a resolution proof

Look at our example again:



and apply resolution rule bottom-up

(Slightly more needed to turn this into formal theorem, but this is essentially it)

CDCL Execution as Resolution Proof

Many more ingredients in modern CDCL SAT solvers [BS97, MS99, MMZ+01], for instance:

- Choice of branching variables crucial
- In leaf, compute & add reason for failure (clause learning)
- Restart every once in a while (saving learned clauses)

CDCL Execution as Resolution Proof

Many more ingredients in modern CDCL SAT solvers [BS97, MS99, MMZ^{+}01], for instance:

- Choice of branching variables crucial
- In leaf, compute & add reason for failure (clause learning)
- Restart every once in a while (saving learned clauses)

But CDCL still yields resolution proofs (though clause learning $\Rightarrow$ general DAGs instead of trees)

Will talk more about this later in the presentation

Resolution Size/Length

Size/length of proof = # clauses (9 in our example)
Length of refuting F = min over all proofs for F

Resolution Size/Length

Size/length of proof = # clauses (9 in our example)
Length of refuting F = min over all proofs for F

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

Some Examples of Hard Formulas w.r.t. Length (1/2)

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

Some Examples of Hard Formulas w.r.t. Length (1/2)

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

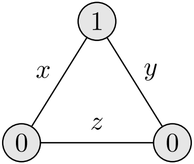
Even onto functional PHP formula is hard for resolution
"Resolution cannot count"

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

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 # true incident edges = label



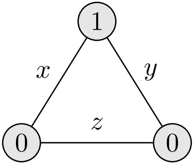
$$\begin{array}{rr} (x \lor y) & \land\ (\overline{x} \lor z) \\ \land\ (\overline{x} \lor \overline{y}) & \land\ (y \lor \overline{z}) \\ \land\ (x \lor \overline{z}) & \land\ (\overline{y} \lor z) \end{array}$$

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

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 # true incident edges = label



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

Requires length $\exp(\Omega(N))$ on well-connected so-called expanders
"Resolution cannot count $\bmod 2$"

Resolution Space

Space $=$ max $\#$ clauses in memory when performing refutation

Motivated by solver memory usage (but also of intrinsical theory interest)

Can be measured in different ways — makes most sense here to focus on clause space

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.	$\perp$	Res$(7,8)$

Resolution Space

Space = max # clauses in memory when performing refutation

Motivated by solver memory usage (but also of intrinsical theory interest)

Can be measured in different ways — makes most sense here to focus on clause space

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

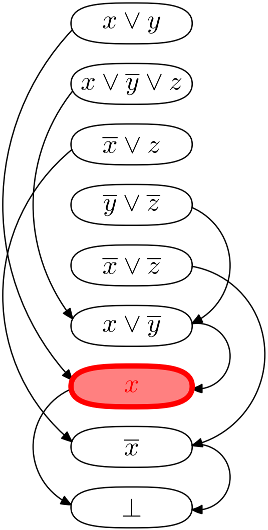
Space = max # clauses in memory when performing refutation

Motivated by solver memory usage (but also of intrinsical theory interest)

Can be measured in different ways — makes most sense here to focus on clause space

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

Example: Space at step 7 . . .



Resolution Space

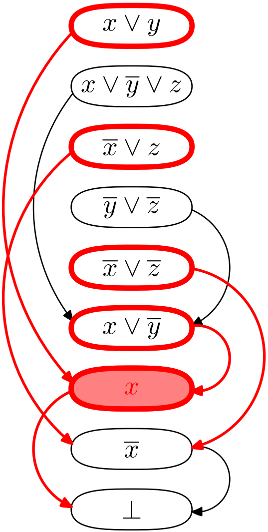
Space = max # clauses in memory when performing refutation

Motivated by solver memory usage (but also of intrinsical theory interest)

Can be measured in different ways — makes most sense here to focus on clause space

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

Example: Space at step 7 is 5



Resolution Space

Space = max # clauses in memory when performing refutation

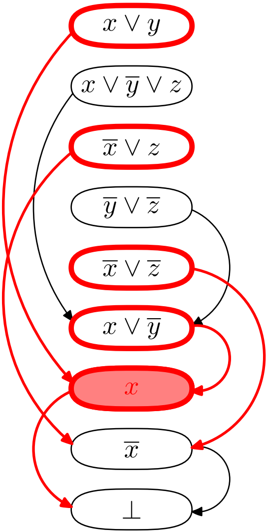
Motivated by solver memory usage (but also of intrinsical theory interest)

Can be measured in different ways — makes most sense here to focus on clause space

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

Example: Space at step 7 is 5

Space of proof = max over all steps



Resolution Space

Space = max # clauses in memory when performing refutation

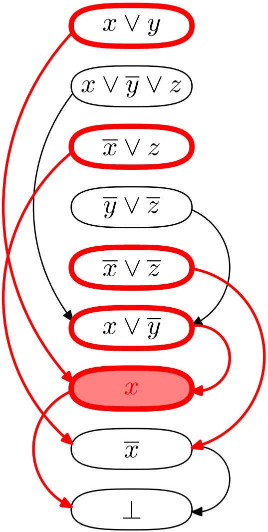
Motivated by solver memory usage (but also of intrinsical theory interest)

Can be measured in different ways — makes most sense here to focus on clause space

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

Example: Space at step 7 is 5

Space of proof = max over all steps
Space of refuting F = min over all proofs



Bounds on Resolution Space

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

Matching $\Omega(N)$ lower bounds known [ABRW02, BG03, ET01]

Bounds on Resolution Space

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

Matching $\Omega(N)$ lower bounds known [ABRW02, BG03, ET01]

Linear space lower bounds might not seem so impressive. . .

Bounds on Resolution Space

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

Matching $\Omega(N)$ lower bounds known [ABRW02, BG03, ET01]

Linear space lower bounds might not seem so impressive. . .

But:

- Apply for space on top of storing formula
- Hold even for optimal algorithms that magically know exactly which clauses to throw away or keep
- So significantly more space might be needed in practice
- And linear space upper bound holds only for proofs of exponential size

Length and Space

Exist space-efficient proofs $\Rightarrow$ exist short proofs [AD08]
(for k-CNF formulas, to be precise)

Length and Space

Exist space-efficient proofs $\Rightarrow$ exist short proofs [AD08]
(for k-CNF formulas, to be precise)

Existence of short proofs $\Rightarrow$ existence of space-efficient proofs?
No!

Length and Space

Exist space-efficient proofs $\Rightarrow$ exist short proofs [AD08]
(for k-CNF formulas, to be precise)

Existence of short proofs $\Rightarrow$ existence of space-efficient proofs?
No!

Pebbling formulas [Nor09, NH13, BN08]

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

Length-Space Trade-offs

Length $\approx$ running time; space $\approx$ memory consumption

SAT solvers aggressively try to minimize both — is this possible?

Length-Space Trade-offs

Length ≈ running time; space ≈ memory consumption
SAT solvers aggressively try to minimize both — is this possible?

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

- Pebbling formulas on the right graphs
- Tseitin formulas on long, narrow rectangular grids

So simultaneous optimization not possible [at least in theory]

Abstract Description of CDCL (1/2)

Trail: a stack of decisions $x_i \stackrel{\mathsf{d}}{=} b$ and unit propagations $x_i \stackrel{C}{=} b$

$$\Big(\underbrace{x_7 \stackrel{\mathsf{d}}{=} 0}_{\text{dec. level 1}}, \underbrace{x_2 \stackrel{\mathsf{d}}{=} 1, x_{12} \stackrel{C_1}{=} 0}_{\text{decision level 2}}, \underbrace{x_6 \stackrel{\mathsf{d}}{=} 1, x_4 \stackrel{C_2}{=} 1, x_1 \stackrel{C_3}{=} 0}_{\text{decision level 3}}, \underbrace{x_{11} \stackrel{\mathsf{d}}{=} 0, x_{59} \stackrel{C_4}{=} 1}_{\text{decision level 4}}\Big)$$

Abstract Description of CDCL (1/2)

Trail: a stack of decisions $x_i \stackrel{\mathsf{d}}{=} b$ and unit propagations $x_i \stackrel{C}{=} b$

$$\Big(\underbrace{x_7 \stackrel{\mathsf{d}}{=} 0}_{\text{dec. level 1}}, \underbrace{x_2 \stackrel{\mathsf{d}}{=} 1, x_{12} \stackrel{C_1}{=} 0}_{\text{decision level 2}}, \underbrace{x_6 \stackrel{\mathsf{d}}{=} 1, x_4 \stackrel{C_2}{=} 1, x_1 \stackrel{C_3}{=} 0}_{\text{decision level 3}}, \underbrace{x_{11} \stackrel{\mathsf{d}}{=} 0, x_{59} \stackrel{C_4}{=} 1}_{\text{decision level 4}}\Big)$$

Clause database $\mathcal{D}$: original formula + learned clauses

Abstract Description of CDCL (1/2)

Trail: a stack of decisions $x_i \stackrel{\mathsf{d}}{=} b$ and unit propagations $x_i \stackrel{C}{=} b$

$$\Big(\underbrace{x_7 \stackrel{\mathsf{d}}{=} 0}_{\text{dec. level 1}}, \underbrace{x_2 \stackrel{\mathsf{d}}{=} 1, x_{12} \stackrel{C_1}{=} 0}_{\text{decision level 2}}, \underbrace{x_6 \stackrel{\mathsf{d}}{=} 1, x_4 \stackrel{C_2}{=} 1, x_1 \stackrel{C_3}{=} 0}_{\text{decision level 3}}, \underbrace{x_{11} \stackrel{\mathsf{d}}{=} 0, x_{59} \stackrel{C_4}{=} 1}_{\text{decision level 4}}\Big)$$

Clause database $\mathcal{D}$: original formula + learned clauses

Start in **Case** mode; transit to **Conflict**, **Unit**, or **Decision**

Abstract Description of CDCL (1/2)

Trail: a stack of decisions $x_i \stackrel{\mathsf{d}}{=} b$ and unit propagations $x_i \stackrel{C}{=} b$

$$\Big(\underbrace{x_7 \stackrel{\mathsf{d}}{=} 0}_{\text{dec. level 1}}, \underbrace{x_2 \stackrel{\mathsf{d}}{=} 1, x_{12} \stackrel{C_1}{=} 0}_{\text{decision level 2}}, \underbrace{x_6 \stackrel{\mathsf{d}}{=} 1, x_4 \stackrel{C_2}{=} 1, x_1 \stackrel{C_3}{=} 0}_{\text{decision level 3}}, \underbrace{x_{11} \stackrel{\mathsf{d}}{=} 0, x_{59} \stackrel{C_4}{=} 1}_{\text{decision level 4}}\Big)$$

Clause database $\mathcal{D}$: original formula + learned clauses

Start in **Case** mode; transit to **Conflict**, **Unit**, or **Decision**

Case **If** trail falsifies clause $C \in \mathcal{D}$, move to **Conflict**;

Abstract Description of CDCL (1/2)

Trail: a stack of decisions $x_i \stackrel{\mathsf{d}}{=} b$ and unit propagations $x_i \stackrel{C}{=} b$

$$\big(\underbrace{x_7 \stackrel{\mathsf{d}}{=} 0}_{\text{dec. level 1}}, \underbrace{x_2 \stackrel{\mathsf{d}}{=} 1, x_{12} \stackrel{C_1}{=} 0}_{\text{decision level 2}}, \underbrace{x_6 \stackrel{\mathsf{d}}{=} 1, x_4 \stackrel{C_2}{=} 1, x_1 \stackrel{C_3}{=} 0}_{\text{decision level 3}}, \underbrace{x_{11} \stackrel{\mathsf{d}}{=} 0, x_{59} \stackrel{C_4}{=} 1}_{\text{decision level 4}}\big)$$

Clause database $\mathcal{D}$: original formula + learned clauses

Start in **Case** mode; transit to **Conflict**, **Unit**, or **Decision**

Case **If** trail falsifies clause $C \in \mathcal{D}$, move to **Conflict**;
else if all variables assigned, output `SAT`;

Abstract Description of CDCL (1/2)

Trail: a stack of decisions $x_i \stackrel{\mathsf{d}}{=} b$ and unit propagations $x_i \stackrel{C}{=} b$

$$(\underbrace{x_7 \stackrel{\mathsf{d}}{=} 0}_{\text{dec. level 1}}, \underbrace{x_2 \stackrel{\mathsf{d}}{=} 1, x_{12} \stackrel{C_1}{=} 0}_{\text{decision level 2}}, \underbrace{x_6 \stackrel{\mathsf{d}}{=} 1, x_4 \stackrel{C_2}{=} 1, x_1 \stackrel{C_3}{=} 0}_{\text{decision level 3}}, \underbrace{x_{11} \stackrel{\mathsf{d}}{=} 0, x_{59} \stackrel{C_4}{=} 1}_{\text{decision level 4}})$$

Clause database $\mathcal{D}$: original formula + learned clauses

Start in **Case** mode; transit to **Conflict**, **Unit**, or **Decision**

Case **If** trail falsifies clause $C \in \mathcal{D}$, move to **Conflict**;
else if all variables assigned, output SAT;
else if some $C \in \mathcal{D}$ unit w.r.t. trail, move to **Unit**;

Abstract Description of CDCL (1/2)

Trail: a stack of decisions $x_i \stackrel{\mathsf{d}}{=} b$ and unit propagations $x_i \stackrel{C}{=} b$

$$\Big(\underbrace{x_7 \stackrel{\mathsf{d}}{=} 0}_{\text{dec. level 1}}, \underbrace{x_2 \stackrel{\mathsf{d}}{=} 1, x_{12} \stackrel{C_1}{=} 0}_{\text{decision level 2}}, \underbrace{x_6 \stackrel{\mathsf{d}}{=} 1, x_4 \stackrel{C_2}{=} 1, x_1 \stackrel{C_3}{=} 0}_{\text{decision level 3}}, \underbrace{x_{11} \stackrel{\mathsf{d}}{=} 0, x_{59} \stackrel{C_4}{=} 1}_{\text{decision level 4}}\Big)$$

Clause database $\mathcal{D}$: original formula + learned clauses

Start in **Case** mode; transit to **Conflict**, **Unit**, or **Decision**

Case **If** trail falsifies clause $C \in \mathcal{D}$, move to **Conflict**;
else if all variables assigned, output `SAT`;
else if some $C \in \mathcal{D}$ unit w.r.t. trail, move to **Unit**;
else if restart, set trail to () and move to **Case**;

Abstract Description of CDCL (1/2)

Trail: a stack of decisions $x_i \stackrel{\mathsf{d}}{=} b$ and unit propagations $x_i \stackrel{C}{=} b$

$$\Big(\underbrace{x_7 \stackrel{\mathsf{d}}{=} 0}_{\text{dec. level 1}}, \underbrace{x_2 \stackrel{\mathsf{d}}{=} 1, x_{12} \stackrel{C_1}{=} 0}_{\text{decision level 2}}, \underbrace{x_6 \stackrel{\mathsf{d}}{=} 1, x_4 \stackrel{C_2}{=} 1, x_1 \stackrel{C_3}{=} 0}_{\text{decision level 3}}, \underbrace{x_{11} \stackrel{\mathsf{d}}{=} 0, x_{59} \stackrel{C_4}{=} 1}_{\text{decision level 4}}\Big)$$

Clause database $\mathcal{D}$: original formula + learned clauses

Start in **Case** mode; transit to **Conflict**, **Unit**, or **Decision**

Case **If** trail falsifies clause $C \in \mathcal{D}$, move to **Conflict**;
else if all variables assigned, output `SAT`;
else if some $C \in \mathcal{D}$ unit w.r.t. trail, move to **Unit**;
else if restart, set trail to () and move to **Case**;
else

1. decide if to apply database reduction to $\mathcal{D}$;
2. move to **Decision**

Abstract Description of CDCL (2/2)

Unit Pick clause $C \in \mathcal{D}$ that is unit w.r.t. trail
(All literals except one is falsified)
Add propagated assignment $x \stackrel{C}{=} b$ to trail
Move to **Case**

Abstract Description of CDCL (2/2)

Unit Pick clause $C \in \mathcal{D}$ that is unit w.r.t. trail
(All literals except one is falsified)
Add propagated assignment $x \stackrel{C}{=} b$ to trail
Move to **Case**

Conflict **If** trail contains no decisions, output `UNSAT`;
else

- apply learning scheme to derive clause C;
- backjump, i.e., remove assignments from trail until C not false but still unit propagates;
- move to **Unit**

Abstract Description of CDCL (2/2)

Unit Pick clause $C \in \mathcal{D}$ that is unit w.r.t. trail
(All literals except one is falsified)
Add propagated assignment $x \stackrel{C}{=} b$ to trail
Move to **Case**

Conflict **If** trail contains no decisions, output `UNSAT`;
else

- apply learning scheme to derive clause C;
- backjump, i.e., remove assignments from trail until C not false but still unit propagates;
- move to **Unit**

Decision Use decision scheme to add decision $x \stackrel{\mathsf{d}}{=} b$ to trail
Move to **Case**

Abstract Description of CDCL (2/2)

Unit Pick clause $C \in \mathcal{D}$ that is unit w.r.t. trail
(All literals except one is falsified)
Add propagated assignment $x \stackrel{C}{=} b$ to trail
Move to **Case**

Conflict **If** trail contains no decisions, output `UNSAT`;
else

- apply learning scheme to derive clause C;
- backjump, i.e., remove assignments from trail until C not false but still unit propagates;
- move to **Unit**

Decision Use decision scheme to add decision $x \stackrel{\mathsf{d}}{=} b$ to trail
Move to **Case**

Description from [EJL+16] drawing heavily on [AFT11, BHJ08, PD11]

CDCL Execution Example

Too small formula for interesting example. . .

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

CDCL Execution Example

Too small formula for interesting example… So expand slightly:

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

CDCL Execution Example

Too small formula for interesting example... So expand slightly:

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

CDCL Execution Example

Too small formula for interesting example... So expand slightly:

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

$w \stackrel{\mathsf{d}}{=} 0$

CDCL Execution Example

Too small formula for interesting example. . . So expand slightly:

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

$w \stackrel{\mathsf{d}}{=} 0$

$u \stackrel{\overline{u} \lor w}{=} 0$

CDCL Execution Example

Too small formula for interesting example. . . So expand slightly:

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

$w \stackrel{\mathsf{d}}{=} 0$

$u \stackrel{\overline{u} \vee w}{=} 0$

$x \stackrel{\mathsf{d}}{=} 0$

CDCL Execution Example

Too small formula for interesting example… So expand slightly:

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

$w \stackrel{\mathsf{d}}{=} 0$

$u \stackrel{\overline{u} \lor w}{=} 0$

$x \stackrel{\mathsf{d}}{=} 0$

$y \stackrel{u \lor x \lor y}{=} 1$

CDCL Execution Example

Too small formula for interesting example... So expand slightly:

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

$w \stackrel{\mathsf{d}}{=} 0$

$u \stackrel{\overline{u} \lor w}{=} 0$

$x \stackrel{\mathsf{d}}{=} 0$

$y \stackrel{u \lor x \lor y}{=} 1$

$z \stackrel{x \lor \overline{y} \lor z}{=} 1$

CDCL Execution Example

Too small formula for interesting example. . . So expand slightly:

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

$w \stackrel{\mathsf{d}}{=} 0$

$u \stackrel{\overline{u} \lor w}{=} 0$

$x \stackrel{\mathsf{d}}{=} 0$

$y \stackrel{u \lor x \lor y}{=} 1$

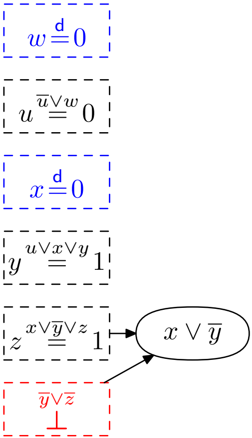
$z \stackrel{x \lor \overline{y} \lor z}{=} 1$

$\stackrel{\overline{y} \lor \overline{z}}{\bot}$

CDCL Execution Example

Too small formula for interesting example... So expand slightly:

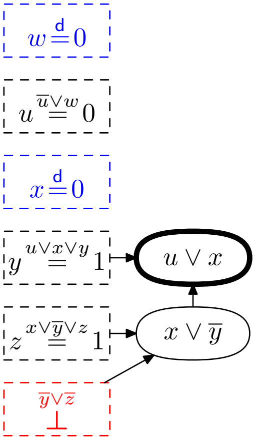
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



CDCL Execution Example

Too small formula for interesting example... So expand slightly:

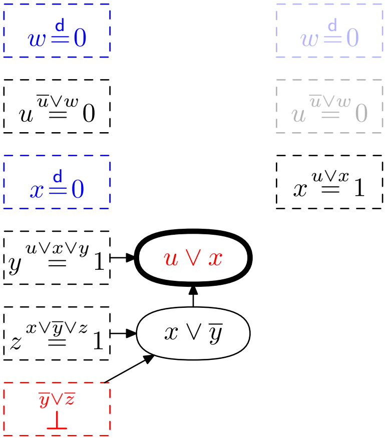
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



CDCL Execution Example

Too small formula for interesting example... So expand slightly:

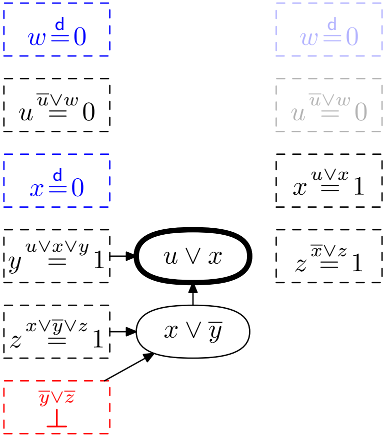
$$(u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{u} \vee w) \wedge (\overline{u} \vee \overline{w})$$



CDCL Execution Example

Too small formula for interesting example... So expand slightly:

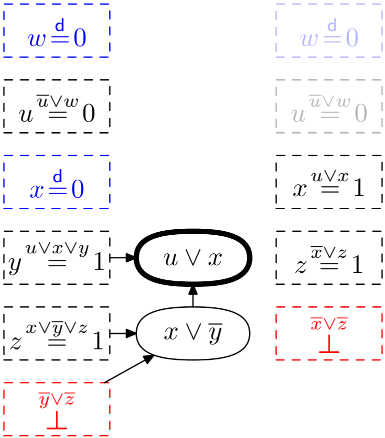
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



CDCL Execution Example

Too small formula for interesting example... So expand slightly:

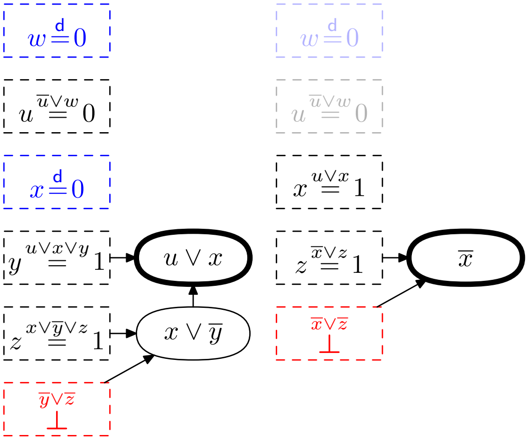
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



CDCL Execution Example

Too small formula for interesting example... So expand slightly:

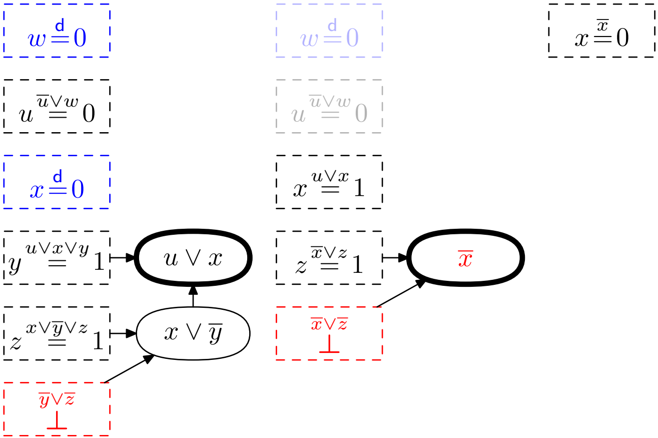
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



CDCL Execution Example

Too small formula for interesting example... So expand slightly:

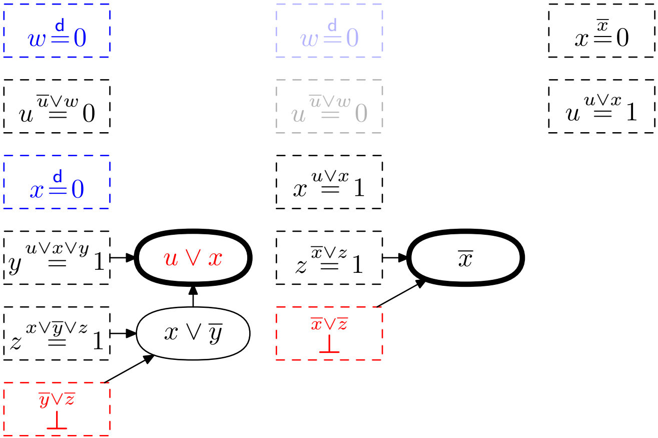
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



CDCL Execution Example

Too small formula for interesting example. . . So expand slightly:

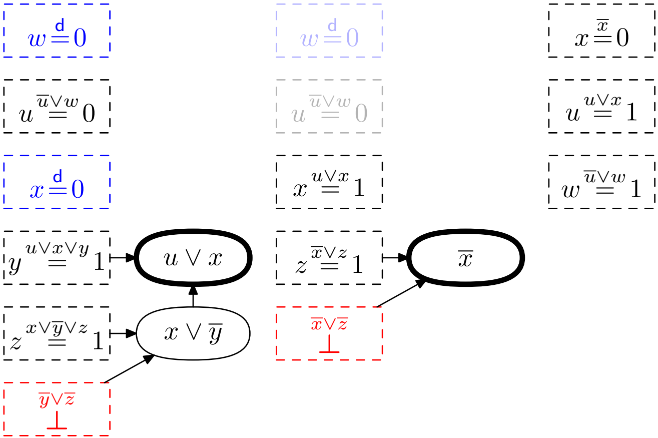
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



CDCL Execution Example

Too small formula for interesting example. . . So expand slightly:

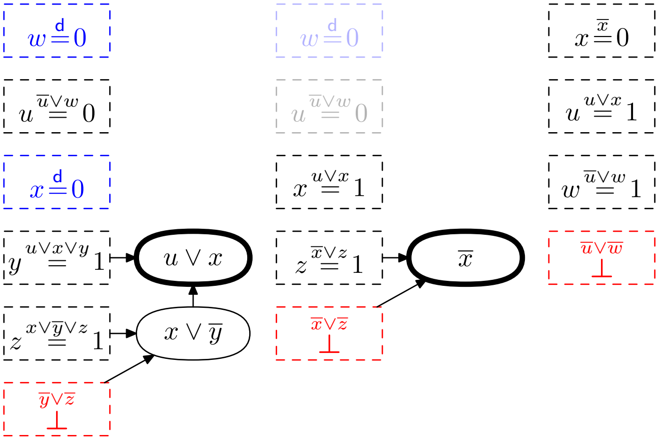
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



CDCL Execution Example

Too small formula for interesting example... So expand slightly:

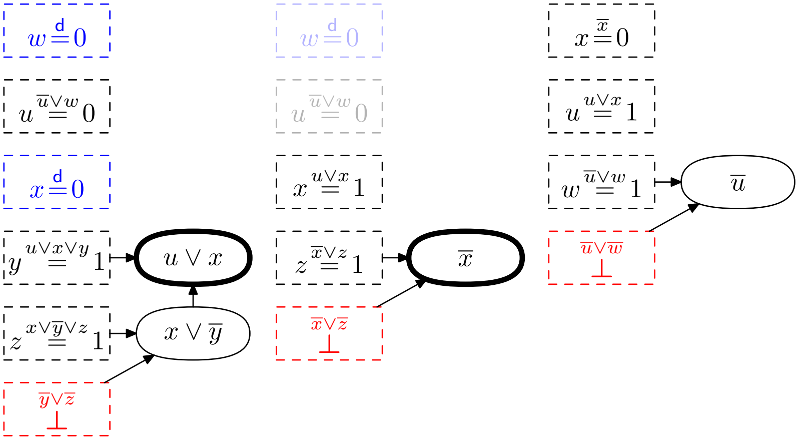
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



CDCL Execution Example

Too small formula for interesting example... So expand slightly:

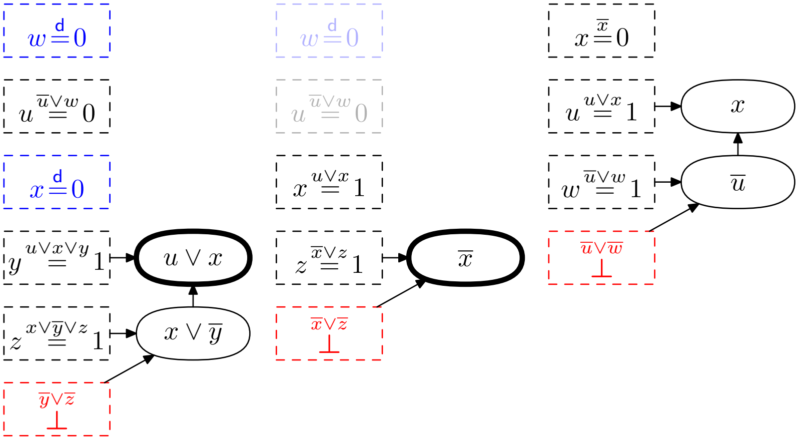
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



CDCL Execution Example

Too small formula for interesting example. . . So expand slightly:

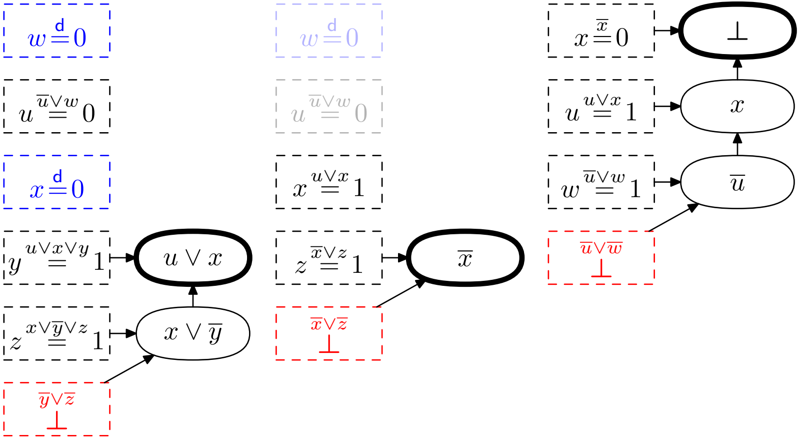
$$(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})$$



CDCL Execution Example

Too small formula for interesting example... So expand slightly:

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

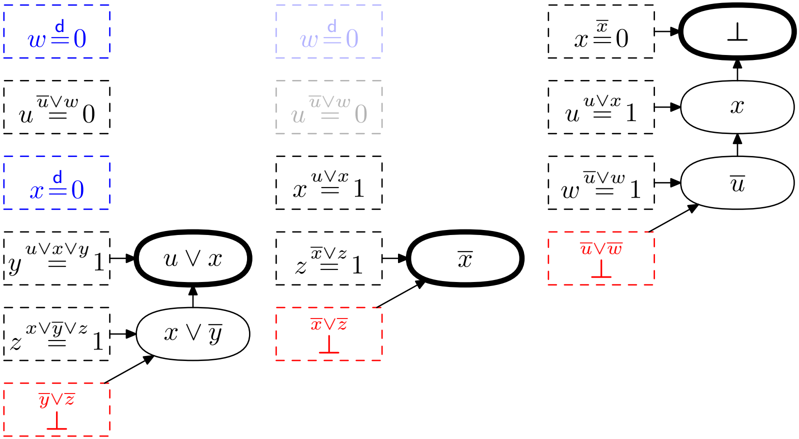


CDCL Execution Example as Resolution Refutation

Obtain resolution refutation. . .

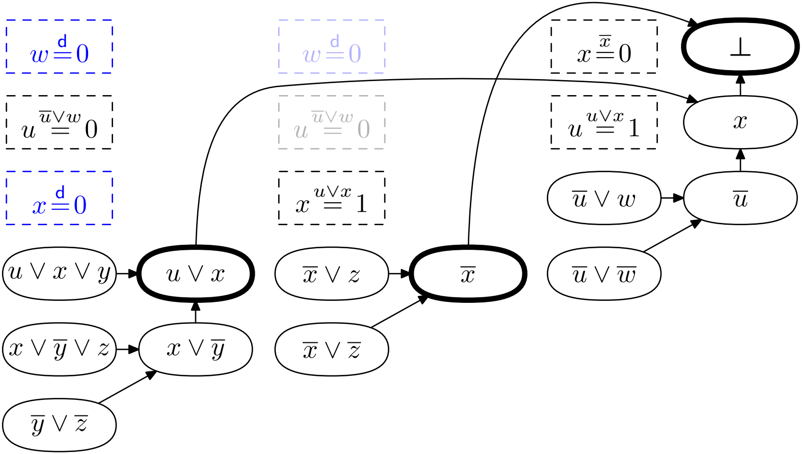
CDCL Execution Example as Resolution Refutation

Obtain resolution refutation from CDCL execution...



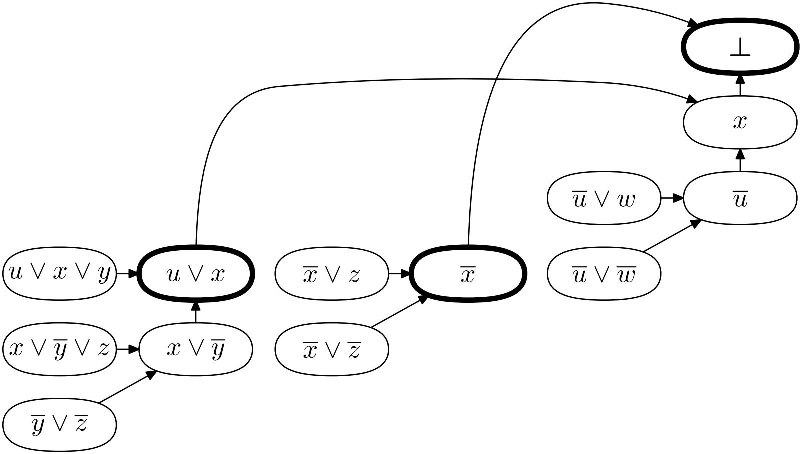
CDCL Execution Example as Resolution Refutation

Obtain resolution refutation from CDCL execution by stringing together conflict analyses:



CDCL Execution Example as Resolution Refutation

Obtain resolution refutation from CDCL execution by stringing together conflict analyses:



Understanding the Efficiency of CDCL Proof Search

- Lower bounds in proof complexity $\Rightarrow$ impossibility results for CDCL even assuming optimal choices

Understanding the Efficiency of CDCL Proof Search

- Lower bounds in proof complexity $\Rightarrow$ impossibility results for CDCL even assuming optimal choices
- But CDCL only finds proofs with very specific structure — can it match resolution upper bounds?

Understanding the Efficiency of CDCL Proof Search

- Lower bounds in proof complexity ⇒ impossibility results for CDCL even assuming optimal choices
- But CDCL only finds proofs with very specific structure — can it match resolution upper bounds?
- Long line of work aimed at proving that CDCL explores resolution search space efficiently, e.g., [BKS04, Van05, BHJ08, HBPV08]

Understanding the Efficiency of CDCL Proof Search

- Lower bounds in proof complexity ⇒ impossibility results for CDCL even assuming optimal choices
- But CDCL only finds proofs with very specific structure — can it match resolution upper bounds?
- Long line of work aimed at proving that CDCL explores resolution search space efficiently, e.g., [BKS04, Van05, BHJ08, HBPV08]
- Challenging problem — progress only by making assumptions such as
  - artificial preprocessing
  - decisions past conflicts
  - non-standard learning scheme
  - no unit propagation(!)

CDCL Simulation of Resolution

General idea is obvious:

- Given resolution proof $(C_1, C_2, \ldots, C_\tau)$
- Force solver to efficiently learn C_t for $t = 1, 2, 3, \ldots$

CDCL Simulation of Resolution

General idea is obvious:

- Given resolution proof $(C_1, C_2, \ldots, C_\tau)$
- Force solver to efficiently learn C_t for $t = 1, 2, 3, \ldots$

Not as easy as it seems. . .

CDCL Simulation of Resolution

General idea is obvious:

- Given resolution proof $(C_1, C_2, \ldots, C_\tau)$
- Force solver to efficiently learn C_t for $t = 1, 2, 3, \ldots$

Not as easy as it seems. . .

- First result in clean model in [PD11]: CDCL as proof system polynomially simulates resolution w.r.t. time/size
- Constructive version in [AFT11]: $\exists$ resolution proof with clauses of bounded size $\Rightarrow$ CDCL will run fast
- Good, so then we're done understanding CDCL?
  Not quite. . .

Room for Further Improvement of [AFT11, PD11]?

- Very frequent **restarts** needed — no progress at all in between
  Restarts are important, but not quite *that* important?!
- **Decision strategy** in [PD11] needs (unknown) resolution proof or should be fully random in [AFT11]
  Probably inherent — fully algorithmic result unlikely [AR08]
- In **clause database** no learned clause must ever be forgotten
  But in practice something like 90–95% of clauses erased...
- Solvers typically have to run in (close to) linear time $\mathcal{O}(n)$
  But **simulation running time** something like $\mathcal{O}(n^5)$

What We Would Want

Want a more fine-grained and realistic CDCL model...

- Capture restarts, clause learning, memory management, etc.
- Modular design to allow study of different features
- Theoretical analogue of projects in [KSM11, SM11, ENSS16]

What We Would Want

Want a more fine-grained and realistic CDCL model. . .

- Capture restarts, clause learning, memory management, etc.
- Modular design to allow study of different features
- Theoretical analogue of projects in [KSM11, SM11, ENSS16]

. . . Leading to improved theoretical insights

- Can CDCL proof search be time *and space* efficient?
- And can it be *really* efficient? (No large polynomial blow-ups)
- How does memory management affect proof search quality?
- Do restarts increase reasoning power?
- How do other heuristics help or hinder proof search?

What We Have So Far (1/2)

- This is ongoing work — reporting results so far in [EJL$^+$16]

What We Have So Far (1/2)

- This is ongoing work — reporting results so far in [EJL+16]
- Much less impressive results than we would have liked...
(but these seem like hard problems)

What We Have So Far (1/2)

- This is ongoing work — reporting results so far in [EJL+16]
- Much less impressive results than we would have liked...
  (but these seem like hard problems)
- Formalize description a few slides back as CDCL proof system

What We Have So Far (1/2)

- This is ongoing work — reporting results so far in [EJL$^+$16]
- Much less impressive results than we would have liked...
  (but these seem like hard problems)
- Formalize description a few slides back as CDCL proof system
- **Proof:** Decisions + conflict analyses + erasures + restarts

What We Have So Far (1/2)

- This is ongoing work — reporting results so far in [EJL+16]
- Much less impressive results than we would have liked...
  (but these seem like hard problems)
- Formalize description a few slides back as CDCL proof system
- **Proof:** Decisions + conflict analyses + erasures + restarts
- **Time/Size:** # decisions + propagations + conflict analysis steps
  Space: (Size of clause database) − (size of formula)

What We Have So Far (2/2)

- Known: no clause learning $\Rightarrow$ collapse to tree-like resolution

What We Have So Far (2/2)

- Known: no clause learning ⇒ collapse to tree-like resolution
- Show too aggressive clause removal ⇒ exponential blow-up in running time, matching theory [BN11, BBI12, BNT13]

What We Have So Far (2/2)

- Known: no clause learning ⇒ collapse to tree-like resolution
- Show too aggressive clause removal ⇒ exponential blow-up in running time, matching theory [BN11, BBI12, BNT13]
- Involves time- and space-efficient CDCL simulations of some resolution proofs (but far from general simulation result)

What We Have So Far (2/2)

- Known: no clause learning ⇒ collapse to tree-like resolution
- Show too aggressive clause removal ⇒ exponential blow-up in running time, matching theory [BN11, BBI12, BNT13]
- Involves time- and space-efficient CDCL simulations of some resolution proofs (but far from general simulation result)
- In addition, these simulations do not need restarts (impossible to prove in principle for model in [AFT11, PD11])

What We Have So Far (2/2)

- Known: no clause learning $\Rightarrow$ collapse to tree-like resolution
- Show too aggressive clause removal $\Rightarrow$ exponential blow-up in running time, matching theory [BN11, BBI12, BNT13]
- Involves time- and space-efficient CDCL simulations of some resolution proofs (but far from general simulation result)
- In addition, these simulations do not need restarts (impossible to prove in principle for model in [AFT11, PD11])
- Intuitively plausible results, but quite painful to formalize

What We Have So Far (2/2)

- Known: no clause learning ⇒ collapse to tree-like resolution
- Show too aggressive clause removal ⇒ exponential blow-up in running time, matching theory [BN11, BBI12, BNT13]
- Involves time- and space-efficient CDCL simulations of some resolution proofs (but far from general simulation result)
- In addition, these simulations do not need restarts (impossible to prove in principle for model in [AFT11, PD11])
- Intuitively plausible results, but quite painful to formalize
- Only math theorems, but have some indications of similar behaviour in practical experiments [ENSS16]

Cutting Planes

Introduced in [CCT87] based on integer LP in [Gom63, Chv73]

Clauses interpreted as linear inequalities over the reals with integer coefficients (identifying $1 \equiv true$ and $0 \equiv false$)

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

Cutting Planes

Introduced in [CCT87] based on integer LP in [Gom63, Chv73]

Clauses interpreted as linear inequalities over the reals with integer coefficients (identifying $1 \equiv true$ and $0 \equiv false$)

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

Derivation rules

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

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

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

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

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

Size, Length and Space

Length = total # lines/inequalities in refutation
Size = sum also size of coefficients
Space = max # lines in memory during refutation

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

Cutting planes

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

Size, Length and Space

Length = total # lines/inequalities in refutation
Size = sum also size of coefficients
Space = max # lines in memory during refutation

Cutting planes

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

Cutting planes

- simulates resolution efficiently w.r.t. length/size and space simultaneously
- is strictly stronger w.r.t. length/size — can refute PHP [CCT87] and subset cardinality formulas efficiently
- 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 an m-clique is not $(m-1)$-colourable"

$p_{i,j}$ = indicator variables for edges in an n-vertex graph
$q_{k,i}$ = identifiers for members of m-clique in graph
$r_{i,\ell}$ = encoding of legal $(m-1)$-colouring of vertices

$q_{k,1} \lor q_{k,2} \lor \cdots \lor q_{k,n}$	some vertex is kth member of clique
$\overline{q}_{k,i} \lor \overline{q}_{k',i}$	clique members are uniquely defined
$p_{i,j} \lor \overline{q}_{k,i} \lor \overline{q}_{k',j}$	clique members are connected by edges
$r_{i,1} \lor r_{i,2} \lor \cdots \lor r_{i,m-1}$	every vertex i has a colour
$\overline{p}_{i,j} \lor \overline{r}_{i,\ell} \lor \overline{r}_{j,\ell}$	neighbours have distinct colours

Exponential lower bound via interpolation and circuit complexity
Technique very specifically tied to structure of formula

Some Challenging Problems for Cutting Planes

Prove length lower bounds for cutting planes

- for Tseitin formulas
- for random k-CNFs

Some Challenging Problems for Cutting Planes

Prove length lower bounds for cutting planes

- for Tseitin formulas
- for random k-CNFs

Prove space lower bounds for cutting planes

- with polynomial-size coefficients (nothing known)
- with constant-size coefficients (very weak bounds in [GPT15])

Some Challenging Problems for Cutting Planes

Prove length lower bounds for cutting planes

- for Tseitin formulas
- for random k-CNFs

Prove space lower bounds for cutting planes

- with polynomial-size coefficients (nothing known)
- with constant-size coefficients (very weak bounds in [GPT15])

Are there trade-offs where the space-efficient CP refutations have small coefficients? (Say, of polynomial or even constant size)

Some Recent News About Cutting Planes

Theorem ([FPPR17, HP17])

Random CNF formulas of logarithmic width are exponentially hard for cutting planes

Some Recent News About Cutting Planes

Theorem ([FPPR17, HP17])

Random CNF formulas of logarithmic width are exponentially hard for cutting planes

Theorem ([dRNV16])

There exist flavours of pebbling formulas such that

- $\exists$ *small-size refutations with constant-size coefficients*
- $\exists$ *small-space refutations with constant-size coefficients*
- *Decreasing the space even for refutations with exponentially large coefficients causes exponential blow-up of length*

What About Conflict-Driven Cutting Planes Solvers?

So-called pseudo-Boolean SAT solvers use (a subset of) cutting planes — but seems hard to make them competitive with CDCL

What About Conflict-Driven Cutting Planes Solvers?

So-called pseudo-Boolean SAT solvers use (a subset of) cutting planes — but seems hard to make them competitive with CDCL

Possible to combine reasoning power of cutting planes with efficiency of CDCL? Work in this direction in, e.g., Sat4j [LP10]

What About Conflict-Driven Cutting Planes Solvers?

So-called pseudo-Boolean SAT solvers use (a subset of) cutting planes — but seems hard to make them competitive with CDCL

Possible to combine reasoning power of cutting planes with efficiency of CDCL? Work in this direction in, e.g., Sat4j [LP10]

Several challenges:

- How detect unit propagation? Not enough to watch just 2 literals (or any finite number)
- Linear constraints more complicated than clauses — and integer arithmetic can become expensive
- Not obvious how to do conflict analysis
  - Can sometimes skip "resolution steps" in conflict analysis with propagating constraints on reason side — good or bad?
  - Can happen that "resolvent" is not conflicting — can be fixed in several ways, but what way is best?

Empirical Analysis of CDCL Solvers

Can we explain empirically when and why CDCL works well (or not)?
Run experiments and draw interesting conclusions?

Empirical Analysis of CDCL Solvers

Can we explain empirically when and why CDCL works well (or not)?
Run experiments and draw interesting conclusions?

- Theory approach: Correlated with complexity measures?
  Some work in [JMNŽ12], but more questions than answers

Empirical Analysis of CDCL Solvers

Can we explain empirically when and why CDCL works well (or not)?
Run experiments and draw interesting conclusions?

- Theory approach: Correlated with complexity measures?
  Some work in [JMNŽ12], but more questions than answers
- Applied approach: Vary settings on industrial benchmarks
  Some work in [KSM11, SM11], but diversity and sparsity of industrial benchmarks makes it hard to draw clear conclusions

Empirical Analysis of CDCL Solvers

Can we explain empirically when and why CDCL works well (or not)?
Run experiments and draw interesting conclusions?

- Theory approach: Correlated with complexity measures?
  Some work in [JMNŽ12], but more questions than answers
- Applied approach: Vary settings on industrial benchmarks
  Some work in [KSM11, SM11], but diversity and sparsity of industrial benchmarks makes it hard to draw clear conclusions

Why not combine the two approaches?

Empirical Analysis of CDCL Solvers

Can we explain empirically when and why CDCL works well (or not)?
Run experiments and draw interesting conclusions?

- Theory approach: Correlated with complexity measures?
  Some work in [JMNŽ12], but more questions than answers
- Applied approach: Vary settings on industrial benchmarks
  Some work in [KSM11, SM11], but diversity and sparsity of industrial benchmarks makes it hard to draw clear conclusions

Why not combine the two approaches?

- Generate scalable & easy versions of theoretical benchmarks
  Have short proofs, so no excuse for solver not doing well. . .

Empirical Analysis of CDCL Solvers

Can we explain empirically when and why CDCL works well (or not)?
Run experiments and draw interesting conclusions?

- Theory approach: Correlated with complexity measures?
  Some work in [JMNŽ12], but more questions than answers
- Applied approach: Vary settings on industrial benchmarks
  Some work in [KSM11, SM11], but diversity and sparsity of industrial benchmarks makes it hard to draw clear conclusions

Why not combine the two approaches?

- Generate scalable & easy versions of theoretical benchmarks
  Have short proofs, so no excuse for solver not doing well...
- Study effect of different CDCL heuristics on performance

Theoretically Easy Combinatorial Benchmarks

- Study tweaked versions of well-studied formulas with:
  - short resolution proofs that can in principle be found by CDCL
  - without any preprocessing
  - often even without any restarts
  - sometimes even without learning, i.e., just DPLL
  - ... given right variable decision order

Theoretically Easy Combinatorial Benchmarks

- Study tweaked versions of well-studied formulas with:
  - short resolution proofs that can in principle be found by CDCL
  - without any preprocessing
  - often even without any restarts
  - sometimes even without learning, i.e., just DPLL
  - ... given right variable decision order
- Test theoretical results in [AFT11, PD11]: Does CDCL search for proofs efficiently?

Theoretically Easy Combinatorial Benchmarks

- Study tweaked versions of well-studied formulas with:
  - short resolution proofs that can in principle be found by CDCL
  - without any preprocessing
  - often even without any restarts
  - sometimes even without learning, i.e., just DPLL
  - . . . given right variable decision order
- Test theoretical results in [AFT11, PD11]: Does CDCL search for proofs efficiently?
- Several benchmarks extremal w.r.t. proof complexity measures or trade-offs — can be expected to "challenge" solver

Instrumented CDCL Solver

To run experiments, add "knobs" to Glucose [AS09, Glu] and vary settings for:

- restart policy
- branching
- clause database management
- clause learning

Instrumented CDCL Solver

To run experiments, add "knobs" to Glucose [AS09, Glu] and vary settings for:

- restart policy
- branching
- clause database management
- clause learning

Yields huge number of potential combinations

- Not all combinations make sense, but many do
- Test also settings where "convential wisdom" knows answer

Some Preliminary Conclusions (1/2)

Importance of restarts

- Sometimes very frequent restarts very important
- Crucial in [AFT11, PD11] for CDCL to simulate resolution efficiently
- Also seems to matter in practice for some formulas which are hard for subsystems of resolution such as regular resolution (stone formulas [AJPU07])

Some Preliminary Conclusions (1/2)

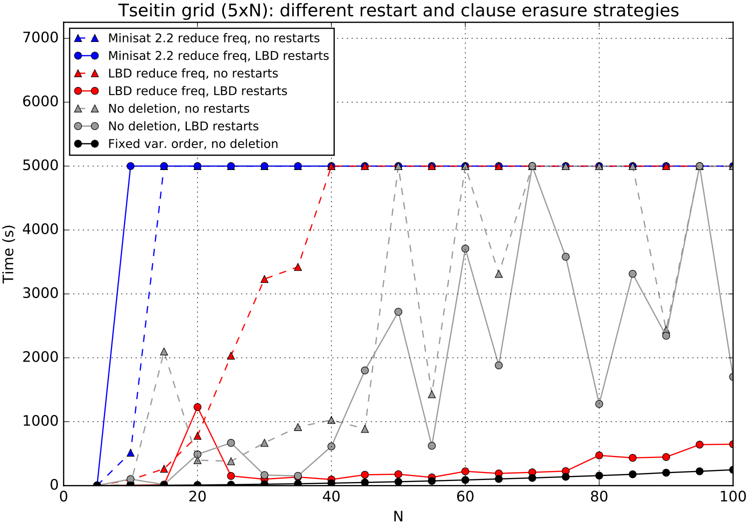
Importance of restarts

- Sometimes very frequent restarts very important
- Crucial in [AFT11, PD11] for CDCL to simulate resolution efficiently
- Also seems to matter in practice for some formulas which are hard for subsystems of resolution such as regular resolution (stone formulas [AJPU07])

Clause erasure

- Theory says very aggressive clause removal could hurt badly
- Seem to see this on scaled-down versions of time-space trade-off formulas in [BBI12, BNT13] (Tseitin formulas)
- Even no erasure at all can be competitive for these formulas for frequent enough restarts

Plot 1: Tseitin Formulas on Grids



Some Preliminary Conclusions (2/2)

Clause assessment

- Can LBD (literal block distance) heuristic balance aggressive erasures by identifying important clauses? Maybe. . .
- But LBD can backfire for too aggressive removal — do old glue clauses clog up the clause database?

Some Preliminary Conclusions (2/2)

Clause assessment

- Can LBD (literal block distance) heuristic balance aggressive erasures by identifying important clauses? Maybe...
- But LBD can backfire for too aggressive removal — do old glue clauses clog up the clause database?

Variable branching

- Variables chosen based on activity in recent conflicts — sometimes small changes in rate of forgetting absolutely crucial (ordering principle formulas [Kri85, Stå96])
- Does slow decay bring solver closer to tree-like resolution???

Some Preliminary Conclusions (2/2)

Clause assessment

- Can LBD (literal block distance) heuristic balance aggressive erasures by identifying important clauses? Maybe. . .
- But LBD can backfire for too aggressive removal — do old glue clauses clog up the clause database?

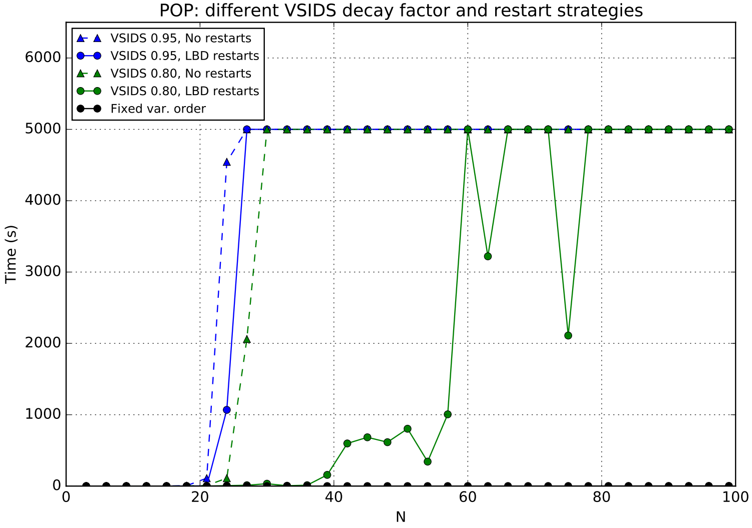
Variable branching

- Variables chosen based on activity in recent conflicts — sometimes small changes in rate of forgetting absolutely crucial (ordering principle formulas [Kri85, Stå96])
- Does slow decay bring solver closer to tree-like resolution???

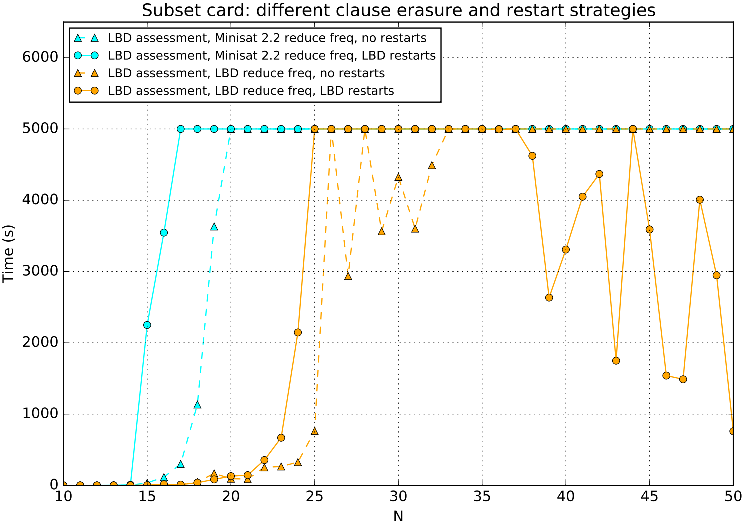
CDCL vs. resolution

- Sometimes CDCL fails miserably on easy formulas
- Sometimes strange easy-hard-easy patterns (subset cardinality formulas [Spe10, VS10, MN14])

Plot 2: Ordering Principle Formulas



Plot 3: Subset Cardinality Formulas



Summing up

This presentation:

- Survey of resolution and connections to CDCL
- Brief discussion of cutting planes and pseudo-Boolean solving
- See survey paper [Nor15] for more details

Summing up

This presentation:

- Survey of resolution and connections to CDCL
- Brief discussion of cutting planes and pseudo-Boolean solving
- See survey paper [Nor15] for more details

Some open problems (not exhaustive list):

- Can CDCL simulate resolution time- and space-efficiently?
- Is standard CDCL without restarts weaker than resolution?
- Can study of subsystems of cutting planes explain power and limitations of pseudo-Boolean solvers?
- Is it possible to build SAT solvers based on stronger proof systems than resolution that beat CDCL solvers?

Summing up

This presentation:

- Survey of resolution and connections to CDCL
- Brief discussion of cutting planes and pseudo-Boolean solving
- See survey paper [Nor15] for more details

Some open problems (not exhaustive list):

- Can CDCL simulate resolution time- and space-efficiently?
- Is standard CDCL without restarts weaker than resolution?
- Can study of subsystems of cutting planes explain power and limitations of pseudo-Boolean solvers?
- Is it possible to build SAT solvers based on stronger proof systems than resolution that beat CDCL solvers?

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*.

[AJPU07] Michael Alekhnovich, Jan Johannsen, Toniann Pitassi, and Alasdair Urquhart. An exponential separation between regular and general resolution. *Theory of Computing*, 3(5):81–102, May 2007. Preliminary version in *STOC '02*.

[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*.

References II

[AS09] Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern SAT solvers. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI '09)*, pages 399–404, July 2009.

[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.

[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*.

[BHJ08] Samuel R. Buss, Jan Hoffmann, and Jan Johannsen. Resolution trees with lemmas: Resolution refinements that characterize DLL-algorithms with clause learning. *Logical Methods in Computer Science*, 4(4:13), December 2008.

References III

[BKS04] Paul Beame, Henry Kautz, and Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning. *Journal of Artificial Intelligence Research*, 22:319–351, December 2004. Preliminary version in *IJCAI '03*.

[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.

References IV

[BS97] Roberto J. Bayardo Jr. and Robert Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97)*, pages 203–208, July 1997.

[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.

[Chv73] Vašek Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4(1):305–337, 1973.

[DLL62] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, July 1962.

[DP60] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.

References V

[dRNV16] Susanna F. de Rezende, Jakob Nordström, and Marc Vinyals. How limited interaction hinders real communication (and what it means for proof and circuit complexity). In *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS '16)*, pages 295–304, October 2016.

[EJL+16] Jan Elffers, Jan Johannsen, Massimo Lauria, Thomas Magnard, Jakob Nordström, and Marc Vinyals. Trade-offs between time and memory in a tighter model of CDCL SAT solvers. In *Proceedings of the 19th International Conference on Theory and Applications of Satisfiability Testing (SAT '16)*, volume 9710 of *Lecture Notes in Computer Science*, pages 160–176. Springer, July 2016.

[ENSS16] Jan Elffers, Jakob Nordström, Laurent Simon, and Karem A. Sakallah. Seeking practical CDCL insights from theoretical SAT benchmarks. Presentation at the *Pragmatics of SAT 2016* workshop. Slides available at `http://www.csc.kth.se/~jakobn/research/TalkPoS16.pdf`, July 2016.

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

References VI

[FPPR17] Noah Fleming, Denis Pankratov, Toniann Pitassi, and Robert Robere. Random $\theta(\log n)$-CNFs are hard for cutting planes. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS '17)*, pages 109–120, October 2017.

[Glu] The Glucose SAT solver. `http://www.labri.fr/perso/lsimon/glucose/`.

[Gom63] Ralph E. Gomory. An algorithm for integer solutions of linear programs. In R.L. Graves and P. Wolfe, editors, *Recent Advances in Mathematical Programming*, pages 269–302. McGraw-Hill, New York, 1963.

[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.

References VII

[HBPV08] Philipp Hertel, Fahiem Bacchus, Toniann Pitassi, and Allen Van Gelder. Clause learning can effectively P-simulate general propositional resolution. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI '08)*, pages 283–290, July 2008.

[HP17] Pavel Hrubeš and Pavel Pudlák. Random formulas, monotone circuits, and interpolation. In *Proceedings of the 58th Annual IEEE Symposium on Foundations of Computer Science (FOCS '17)*, pages 121–131, October 2017.

[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.

[Kri85] Balakrishnan Krishnamurthy. Short proofs for tricky formulas. *Acta Informatica*, 22(3):253–275, August 1985.

References VIII

[KSM11] Hadi Katebi, Karem A. Sakallah, and João P. Marques-Silva. Empirical study of the anatomy of modern SAT solvers. In *Proceedings of the 14th International Conference on Theory and Applications of Satisfiability Testing (SAT '11)*, volume 6695 of *Lecture Notes in Computer Science*, pages 343–356. Springer, June 2011.

[LP10] Daniel Le Berre and Anne Parrain. The Sat4j library, release 2.2. *Journal on Satisfiability, Boolean Modeling and Computation*, 7:59–64, 2010.

[MMZ+01] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference (DAC '01)*, pages 530–535, June 2001.

[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.

References IX

[MS99] João P. Marques-Silva and Karem A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, May 1999. Preliminary version in *ICCAD '96*.

[NH13] Jakob Nordström and Johan Håstad. Towards an optimal separation of space and length in resolution. *Theory of Computing*, 9:471–557, May 2013. Preliminary version in *STOC '08*.

[Nor09] Jakob Nordström. Narrow proofs may be spacious: Separating space and width in resolution. *SIAM Journal on Computing*, 39(1):59–121, May 2009. Preliminary version in *STOC '06*.

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

[PD11] Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning SAT solvers as resolution engines. *Artificial Intelligence*, 175(2):512–525, February 2011. Preliminary version in *CP '09*.

References X

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

[SM11] Karem A. Sakallah and João Marques-Silva. Anatomy and empirical evaluation of modern SAT solvers. *Bulletin of the European Association for Theoretical Computer Science*, 103:96–121, February 2011.

[Spe10] Ivor Spence. sgen1: A generator of small but difficult satisfiability benchmarks. *Journal of Experimental Algorithmics*, 15:1.2:1–1.2:15, March 2010.

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

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

References XI

[Van05] Allen Van Gelder. Pool resolution and its relation to regular resolution and DPLL with clause learning. In *Proceedings of the 12th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR '05)*, volume 3835 of *Lecture Notes in Computer Science*, pages 580–594. Springer, 2005.

[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.