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

Jakob Nordström

KTH Royal Institute of Technology
Stockholm, Sweden

Theoretical Foundations of SAT Solving
Fields Institute, Toronto, Canada
August 15–19, 2016
Understanding Conflict-Driven SAT Solving Through the Lens of Proof Complexity?

Jakob Nordström

KTH Royal Institute of Technology
Stockholm, Sweden

Theoretical Foundations of SAT Solving
Fields Institute, Toronto, Canada
August 15–19, 2016
The unreasonable effectiveness of SAT solvers

- The Boolean satisfiability problem (SAT) is NP-complete and so should be exponentially hard.
- Yet current state-of-the-art 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?
The unreasonable effectiveness of SAT solvers

- The Boolean satisfiability problem (SAT) is NP-complete and so should be exponentially hard
- Yet current state-of-the-art 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?
The unreasonable effectiveness of SAT solvers

- The Boolean satisfiability problem (SAT) is NP-complete and so should be exponentially hard
- Yet current state-of-the-art 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
The unreasonable effectiveness of SAT solvers

- The Boolean satisfiability problem (SAT) is NP-complete and so should be exponentially hard.
- Yet current state-of-the-art 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
The unreasonable effectiveness of SAT solvers

- The Boolean satisfiability problem (SAT) is NP-complete and so should be exponentially hard.
- Yet current state-of-the-art 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 (including some ongoing work)
- Show some theoretical “benchmark formulas” used to understand potential and limitations of SAT solvers
- Discuss some (of the many) remaining open problems
Purpose of This Presentation

- Survey some of the research in the area (including some ongoing work)
- Show some theoretical “benchmark formulas” used to understand potential and limitations of SAT solvers
- Discuss some (of the many) remaining open problems

Caveats:
- By necessity, selective and somewhat subjective coverage
- Won’t do too much name-dropping — full references at end of slides
Some More Caveats and Clarifications

Only basic propositional logic proof search
- No SMT or first-order logic or anything in this talk
- No discussion of preprocessing techniques

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 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)
Some More Caveats and Clarifications

Only basic propositional logic proof search
- No SMT or first-order logic or anything in this talk
- No discussion of preprocessing techniques

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

Only basic propositional logic proof search
- No SMT or first-order logic or anything in this talk
- No discussion of preprocessing techniques

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 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)
1 Resolution and Conflict-Driven Clause Learning
   - The Resolution Proof System
   - Conflict-Driven Clause Learning
   - Theoretical Analysis of CDCL

2 Cutting Planes and Pseudo-Boolean SAT Solving
   - The Cutting Planes Proof System
   - Pseudo-Boolean SAT Solving

3 Seeking Practical CDCL Insights from Theoretical Benchmarks
   - Experimental Set-up
   - Some Tentative Findings
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)

- **\( O(f(N)) \)** 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**

\[
\frac{C \lor x \quad D \lor \overline{x}}{
\quad C \lor D}
\]

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**

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

Done when empty clause $\bot$ derived

<table>
<thead>
<tr>
<th>Step</th>
<th>Clause</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>$x \lor y$</td>
</tr>
<tr>
<td>2.</td>
<td>$x \lor \overline{y} \lor z$</td>
</tr>
<tr>
<td>3.</td>
<td>$\overline{x} \lor z$</td>
</tr>
<tr>
<td>4.</td>
<td>$\overline{y} \lor \overline{z}$</td>
</tr>
<tr>
<td>5.</td>
<td>$\overline{x} \lor \overline{z}$</td>
</tr>
</tbody>
</table>
### The Resolution Proof System Underlying CDCL

**Goal:** refute **unsatisfiable** CNF

**Start with clauses of formula (axioms)**

**Derive new clauses by resolution rule**

\[
\frac{C \lor x \quad D \lor \overline{x}}{C \lor D}
\]

**Done when empty clause \(\perp\) derived**

**Can represent refutation-proof as**
- annotated list or
- directed acyclic graph

<table>
<thead>
<tr>
<th>Step</th>
<th>Clause</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>(x \lor y)</td>
<td>Axiom</td>
</tr>
<tr>
<td>2.</td>
<td>(x \lor \overline{y} \lor z)</td>
<td>Axiom</td>
</tr>
<tr>
<td>3.</td>
<td>(\overline{x} \lor z)</td>
<td>Axiom</td>
</tr>
<tr>
<td>4.</td>
<td>(\overline{y} \lor \overline{z})</td>
<td>Axiom</td>
</tr>
<tr>
<td>5.</td>
<td>(\overline{x} \lor \overline{z})</td>
<td>Axiom</td>
</tr>
<tr>
<td>6.</td>
<td>(x \lor \overline{y})</td>
<td>Res(2, 4)</td>
</tr>
<tr>
<td>7.</td>
<td>(x)</td>
<td>Res(1, 6)</td>
</tr>
<tr>
<td>8.</td>
<td>(\overline{x})</td>
<td>Res(3, 5)</td>
</tr>
<tr>
<td>9.</td>
<td>(\perp)</td>
<td>Res(7, 8)</td>
</tr>
</tbody>
</table>
The Resolution Proof System Underlying CDCL

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by **resolution rule**

\[
\frac{C \lor x}{C \lor D} \quad \frac{D \lor \bar{x}}{C \lor D}
\]

Done when empty clause $\bot$ derived

Can represent refutation-proof as

- annotated list or
- directed acyclic graph

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

\[
\frac{C \lor x}{C \lor D} \quad \frac{D \lor \overline{x}}{C \lor D}
\]

Done when empty clause \( \bot \) derived

Can represent refutation/proof as

- annotated list or
- directed acyclic graph

<table>
<thead>
<tr>
<th>Step</th>
<th>Clause</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>( x \lor y )</td>
<td>Axiom</td>
</tr>
<tr>
<td>2.</td>
<td>( x \lor \overline{y} \lor z )</td>
<td>Axiom</td>
</tr>
<tr>
<td>3.</td>
<td>( \overline{x} \lor z )</td>
<td>Axiom</td>
</tr>
<tr>
<td>4.</td>
<td>( \overline{y} \lor \overline{z} )</td>
<td>Axiom</td>
</tr>
<tr>
<td>5.</td>
<td>( \overline{x} \lor \overline{z} )</td>
<td>Axiom</td>
</tr>
<tr>
<td>6.</td>
<td>( x \lor \overline{y} )</td>
<td>Res((2, 4))</td>
</tr>
<tr>
<td>7.</td>
<td>( x )</td>
<td>Res((1, 6))</td>
</tr>
<tr>
<td>8.</td>
<td>( \overline{x} )</td>
<td>Res((3, 5))</td>
</tr>
<tr>
<td>9.</td>
<td>( \bot )</td>
<td>Res((7, 8))</td>
</tr>
</tbody>
</table>
The Resolution Proof System Underlying CDCL

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

\[
\frac{C \lor x}{C \lor D} \quad \frac{D \lor \overline{x}}{C \lor D}
\]

Done when empty clause \( \bot \) derived

Can represent refutation/proof as

- annotated list or
- directed acyclic graph

<table>
<thead>
<tr>
<th>Step</th>
<th>Clause</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>( x \lor y )</td>
<td>Axiom</td>
</tr>
<tr>
<td>2.</td>
<td>( x \lor \overline{y} \lor z )</td>
<td>Axiom</td>
</tr>
<tr>
<td>3.</td>
<td>( \overline{x} \lor z )</td>
<td>Axiom</td>
</tr>
<tr>
<td>4.</td>
<td>( \overline{y} \lor \overline{z} )</td>
<td>Axiom</td>
</tr>
<tr>
<td>5.</td>
<td>( \overline{x} \lor \overline{z} )</td>
<td>Axiom</td>
</tr>
<tr>
<td>6.</td>
<td>( x \lor \overline{y} )</td>
<td>Res(2, 4)</td>
</tr>
<tr>
<td>7.</td>
<td>( x )</td>
<td>Res(1, 6)</td>
</tr>
<tr>
<td>8.</td>
<td>( \overline{x} )</td>
<td>Res(3, 5)</td>
</tr>
<tr>
<td>9.</td>
<td>( \bot )</td>
<td>Res(7, 8)</td>
</tr>
</tbody>
</table>
The Resolution Proof System Underlying CDCL

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (**axioms**)  

Derive new clauses by **resolution rule**

\[
\begin{array}{c}
C \lor x \\
\hline
D \lor \overline{x}
\end{array}
\]

\[
\frac{C \lor x \quad D \lor \overline{x}}{C \lor D}
\]

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

\[
\frac{C \lor x \quad D \lor \overline{x}}{C \lor D}
\]

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**)  
\[
\begin{align*}
&1. \quad x \lor y \\
&2. \quad x \lor \overline{y} \lor z
\end{align*}
\]

Derive new clauses by **resolution rule**  
\[
\frac{C \lor x \quad D \lor \overline{x}}{C \lor D}
\]

Done when empty clause \( \bot \) derived  
\[
\begin{align*}
&3. \quad \overline{x} \lor z \\
&4. \quad \overline{y} \lor \overline{z} \\
&5. \quad \overline{x} \lor \overline{z} \\
&6. \quad x \lor \overline{y} \\
&7. \quad x \\
&8. \quad \overline{x} \\
&9. \quad \bot
\end{align*}
\]

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

\[
\frac{C \lor x}{C \lor D} \quad \frac{D \lor \bar{x}}{C \lor D}
\]

Done when empty clause $\bot$ derived

Can represent refutation/proof as

- annotated list or
- directed acyclic graph

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

\[ \frac{C \lor x}{C \lor D} \]

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)*

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>( x \lor y )</td>
<td>Axiom</td>
</tr>
<tr>
<td>2.</td>
<td>( x \lor \overline{y} \lor z )</td>
<td>Axiom</td>
</tr>
</tbody>
</table>

### Derive new clauses by **resolution rule**

\[
\frac{C \lor x \quad D \lor \overline{x}}{C \lor D}
\]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3.</td>
<td>( \overline{x} \lor z )</td>
<td>Axiom</td>
</tr>
<tr>
<td>4.</td>
<td>( \overline{y} \lor \overline{z} )</td>
<td>Axiom</td>
</tr>
<tr>
<td>5.</td>
<td>( \overline{x} \lor \overline{z} )</td>
<td>Axiom</td>
</tr>
</tbody>
</table>

### Done when empty clause \( \bot \) derived

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6.</td>
<td>( x \lor \overline{y} )</td>
<td>Res((2, 4))</td>
</tr>
<tr>
<td>7.</td>
<td>( x )</td>
<td>Res((1, 6))</td>
</tr>
<tr>
<td>8.</td>
<td>( \overline{x} )</td>
<td>Res((3, 5))</td>
</tr>
<tr>
<td>9.</td>
<td>( \bot )</td>
<td>Res((7, 8))</td>
</tr>
</tbody>
</table>
### The Resolution Proof System Underlying CDCL

**Goal:** refute **unsatisfiable** CNF

**Start with clauses of formula (axioms)**

**Derive new clauses by resolution rule**

\[
\frac{C \lor x}{C \lor D} \quad \frac{D \lor \overline{x}}{C \lor \overline{x}}
\]

**Done when empty clause \( \bot \) derived**

**Can represent refutation/proof as**

- annotated list or
- directed acyclic graph

<table>
<thead>
<tr>
<th>Step</th>
<th>Clause</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>( x \lor y )</td>
<td>Axiom</td>
</tr>
<tr>
<td>2.</td>
<td>( x \lor \overline{y} \lor z )</td>
<td>Axiom</td>
</tr>
<tr>
<td>3.</td>
<td>( \overline{x} \lor z )</td>
<td>Axiom</td>
</tr>
<tr>
<td>4.</td>
<td>( \overline{y} \lor \overline{z} )</td>
<td>Axiom</td>
</tr>
<tr>
<td>5.</td>
<td>( \overline{x} \lor \overline{z} )</td>
<td>Axiom</td>
</tr>
<tr>
<td>6.</td>
<td>( x \lor \overline{y} )</td>
<td>Res(2, 4)</td>
</tr>
<tr>
<td>7.</td>
<td>( x )</td>
<td>Res(1, 6)</td>
</tr>
<tr>
<td>8.</td>
<td>( \overline{x} )</td>
<td>Res(3, 5)</td>
</tr>
<tr>
<td>9.</td>
<td>( \bot )</td>
<td>Res(7, 8)</td>
</tr>
</tbody>
</table>
The Resolution Proof System Underlying CDCL

<table>
<thead>
<tr>
<th>Step</th>
<th>Clause</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>$x \lor y$</td>
<td>Axiom</td>
</tr>
<tr>
<td>2.</td>
<td>$x \lor \overline{y} \lor z$</td>
<td>Axiom</td>
</tr>
<tr>
<td>3.</td>
<td>$\overline{x} \lor z$</td>
<td>Axiom</td>
</tr>
<tr>
<td>4.</td>
<td>$\overline{y} \lor \overline{z}$</td>
<td>Axiom</td>
</tr>
<tr>
<td>5.</td>
<td>$\overline{x} \lor \overline{z}$</td>
<td>Axiom</td>
</tr>
<tr>
<td>6.</td>
<td>$x \lor \overline{y}$</td>
<td>Res$(2, 4)$</td>
</tr>
<tr>
<td>7.</td>
<td>$x$</td>
<td>Res$(1, 6)$</td>
</tr>
<tr>
<td>8.</td>
<td>$\overline{x}$</td>
<td>Res$(3, 5)$</td>
</tr>
<tr>
<td>9.</td>
<td>$\bot$</td>
<td>Res$(7, 8)$</td>
</tr>
</tbody>
</table>

Goal: refute **unsatisfiable** CNF

Start with clauses of formula *axioms*

Derive new clauses by **resolution rule**

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

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**

\[
\begin{array}{c}
\frac{C \lor x}{\frac{D \lor \overline{x}}{C \lor D}} \\
\end{array}
\]

Done when empty clause \( \bot \) derived

Can represent refutation/proof as

- annotated list or
- directed acyclic graph

<table>
<thead>
<tr>
<th>Step</th>
<th>Clause</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( x \lor y )</td>
<td>Axiom</td>
</tr>
<tr>
<td>2</td>
<td>( x \lor \overline{y} \lor z )</td>
<td>Axiom</td>
</tr>
<tr>
<td>3</td>
<td>( \overline{x} \lor z )</td>
<td>Axiom</td>
</tr>
<tr>
<td>4</td>
<td>( \overline{y} \lor \overline{z} )</td>
<td>Axiom</td>
</tr>
<tr>
<td>5</td>
<td>( \overline{x} \lor \overline{z} )</td>
<td>Axiom</td>
</tr>
<tr>
<td>6</td>
<td>( x \lor \overline{y} )</td>
<td>Res(2, 4)</td>
</tr>
<tr>
<td>7</td>
<td>( x )</td>
<td>Res(1, 6)</td>
</tr>
<tr>
<td>8</td>
<td>( \overline{x} )</td>
<td>Res(3, 5)</td>
</tr>
<tr>
<td>9</td>
<td>( \bot )</td>
<td>Res(7, 8)</td>
</tr>
</tbody>
</table>
The Resolution Proof System Underlying CDCL

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (**axioms**)  

Derive new clauses by **resolution rule**

\[
\frac{C \lor x \quad D \lor \overline{x}}{C \lor D}
\]

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**

\[
\frac{C \lor x \quad D \lor \overline{x}}{C \lor D}
\]

Done when empty clause \( \bot \) derived

Can represent refutation/proof as
- annotated list or
- **directed acyclic graph**

**Tree-like resolution** if DAG is tree
The Resolution Proof System Underlying CDCL

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

\[
\frac{C \lor x \quad D \lor \lnot x}{C \lor D}
\]

Done when empty clause \(\bot\) derived

Can represent refutation/proof as
- annotated list or
- directed acyclic graph

Tree-like resolution if DAG is tree
**Regular** if resolved variables don’t repeat on path
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 falsified clause found

\[
\begin{align*}
    &x \lor y \\
    &y \\
    &z \\
    &x \lor \neg y \lor z \\
    &\neg y \lor \neg z
\end{align*}
\]
DPLL Execution as Resolution Proof

A DPLL execution is essentially a resolution proof
DPLL Execution as Resolution Proof

A DPLL execution is essentially a resolution proof

Look at our example again:

\[
\begin{align*}
  x \lor y \\
  y \\
  x \lor y \\
  z \\
  x \lor y \lor z \\
  \overline{y} \lor \overline{z} \\
\end{align*}
\]
DPLL Execution as Resolution Proof

A DPLL execution is essentially a resolution proof

Look at our example again:

\[
\begin{array}{c}
x \lor y \\
0 \\
\end{array}
\begin{array}{c}
y \\
1 \\
\end{array}
\begin{array}{c}
x \lor y \\
1 \\
\end{array}
\begin{array}{c}
z \\
0 \\
\end{array}
\begin{array}{c}
x \lor z \\
0 \\
\end{array}
\begin{array}{c}
\overline{x} \lor z \\
1 \\
\end{array}
\begin{array}{c}
x \lor z \\
1 \\
\end{array}
\begin{array}{c}
\overline{x} \lor \overline{z} \\
0 \\
\end{array}
\begin{array}{c}
\overline{x} \lor \overline{z} \\
1 \\
\end{array}
\end{array}
\]

and apply resolution rule bottom-up
DPLL Execution as Resolution Proof

A DPLL execution is essentially a resolution proof

Look at our example again:

and apply resolution rule bottom-up
DPLL Execution as Resolution Proof

A DPLL execution is essentially a resolution proof

Look at our example again:

\[
\begin{align*}
&x \
&x \vee y \
&x \vee \overline{y} \
&x \vee \overline{y} \vee z \
&\overline{x} \vee z \
&\overline{x} \vee \overline{z} \\
\end{align*}
\]

and apply resolution rule bottom-up
A DPLL execution is essentially a resolution proof

Look at our example again:

\[
\begin{align*}
\neg x \lor y \\
\neg x \lor \neg y \\
\neg x \lor z \\
\neg x \lor \neg z \\
x \lor \neg y \lor z \\
\neg y \lor \neg z
\end{align*}
\]

and apply resolution rule bottom-up
A DPLL execution is essentially a resolution proof

Look at our example again:

\[
\begin{align*}
  x & \lor y \\
  x & \lor \overline{y} \\
  x & \lor \overline{y} \lor z \\
  x & \lor \overline{y} \lor \overline{z} \\
  \overline{x} & \lor z \\
  \overline{x} & \lor \overline{z} \\
  \overline{x} & \lor \overline{z} \\
  \perp &
\end{align*}
\]

and apply resolution rule bottom-up
A DPLL execution is essentially 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)
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)
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 ⇒ general DAGs instead of trees)
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 $= \# \text{ clauses}$ (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 $\exp(O(N))$

Matching $\exp(\Omega(N))$ lower bounds known
Some Examples of Hard Formulas w.r.t. 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 $j$”

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

Can also add “functionality” and “onto” axioms

\[
\overline{p}_{i,j} \lor \overline{p}_{i,j'} \quad \text{no pigeon } i \text{ gets two holes } j \neq j'
\]
\[
p_{1,j} \lor p_{2,j} \lor \cdots \lor p_{n+1,j} \quad \text{every hole } j \text{ gets a pigeon}
\]
Some Examples of Hard Formulas w.r.t. Length (1/3)

**Pigeonhole principle (PHP)** [Hak85]

“$n + 1$ pigeons don’t fit into $n$ holes”

Variables $p_{i,j} = \text{“pigeon } i \text{ goes into hole } j\text{”}$

\[
p_{i,1} \lor p_{i,2} \lor \cdots \lor p_{i,n} \quad \text{every pigeon } i \text{ gets a hole}
\]

\[
\overline{p}_{i,j} \lor \overline{p}_{i',j} \quad \text{no hole } j \text{ gets two pigeons } i \neq i'
\]

Can also add “functionality” and “onto” axioms

\[
\overline{p}_{i,j} \lor \overline{p}_{i,j'} \quad \text{no pigeon } i \text{ gets two holes } j \neq j'
\]

\[
p_{1,j} \lor p_{2,j} \lor \cdots \lor p_{n+1,j} \quad \text{every hole } j \text{ gets a pigeon}
\]

Even onto functional PHP formula is hard for resolution

“Resolution cannot count”
Some Examples of Hard Formulas w.r.t. 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 $j$”

\[
p_{i,1} \lor p_{i,2} \lor \cdots \lor p_{i,n} \quad \text{every pigeon } i \text{ gets a hole}
\]

\[
\overline{p}_{i,j} \lor \overline{p}_{i',j} \quad \text{no hole } j \text{ gets two pigeons } i \neq i'
\]

Can also add “functionality” and “onto” axioms

\[
\overline{p}_{i,j} \lor \overline{p}_{i,j'} \quad \text{no pigeon } i \text{ gets two holes } j \neq j'
\]

\[
p_{1,j} \lor p_{2,j} \lor \cdots \lor p_{n+1,j} \quad \text{every hole } j \text{ gets a pigeon}
\]

Even onto functional PHP formula is hard for resolution

“Resolution cannot count”

But only length lower bound $\exp(\Omega(\sqrt[3]{N}))$ in terms of formula size
Some Examples of Hard Formulas w.r.t. Length (2/3)

**Tseitin formulas** [Urquhart 87]
“Sum of degrees of vertices in graph is even”

Variables = edges (in undirected graph of bounded degree)

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

```
(x ∨ y) ∧ (x ∨ z) ∧ (x ∨ z) ∧ (y ∨ z) ∧ (x ∨ z) ∧ (y ∨ z)
```

![Graph Diagram](image-url)
Some Examples of Hard Formulas w.r.t. 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 \# true incident edges = label

\[
\begin{align*}
(x \lor y) & \land (\overline{x} \lor z) \\
\land (\overline{x} \lor \overline{y}) & \land (y \lor \overline{z}) \\
\land (x \lor \overline{z}) & \land (\overline{y} \lor z)
\end{align*}
\]

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

“Resolution cannot count \text{ mod } 2”
Some Examples of Hard Formulas w.r.t. Length (3/3)

**Subset cardinality formulas** [Spe10, VS10, MN14]

Variables = 1s in matrix with four 1s per row/column + extra 1
Row ⇒ majority of variables true; column ⇒ majority false

\[
\begin{pmatrix}
1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1
\end{pmatrix}
\land (x_{1,1} \lor x_{1,2} \lor x_{1,4})
\land (x_{1,1} \lor x_{1,2} \lor x_{1,8})
\land (x_{1,1} \lor x_{1,4} \lor x_{1,8})
\land (x_{1,2} \lor x_{1,4} \lor x_{1,8})
\land (\overline{x}_{4,11} \lor \overline{x}_{8,11} \lor \overline{x}_{10,11})
\land (\overline{x}_{4,11} \lor \overline{x}_{8,11} \lor \overline{x}_{11,11})
\land (\overline{x}_{4,11} \lor \overline{x}_{10,11} \lor \overline{x}_{11,11})
\land (\overline{x}_{8,11} \lor \overline{x}_{10,11} \lor \overline{x}_{11,11})\]

Jakob Nordström (KTH)  
Understanding CDCL Through Lens of Proof Complexity  
Fields Institute Aug ’16 14/51
Some Examples of Hard Formulas w.r.t. Length (3/3)

**Subset cardinality formulas** [Spe10, VS10, MN14]

Variables = 1s in matrix with four 1s per row/column \(\textbf{+ extra} \ 1\)
Row \(\Rightarrow\) majority of variables true; column \(\Rightarrow\) majority false

\[
\begin{pmatrix}
1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
\end{pmatrix}
\]

\[
(x_{1,1} \lor x_{1,2} \lor x_{1,4}) \\
\land (x_{1,1} \lor x_{1,2} \lor x_{1,8}) \\
\land (x_{1,1} \lor x_{1,4} \lor x_{1,8}) \\
\land (x_{1,2} \lor x_{1,4} \lor x_{1,8}) \\
\vdots \\
\land (\overline{x}_{4,11} \lor \overline{x}_{8,11} \lor \overline{x}_{10,11}) \\
\land (\overline{x}_{4,11} \lor \overline{x}_{8,11} \lor \overline{x}_{11,11}) \\
\land (\overline{x}_{4,11} \lor \overline{x}_{10,11} \lor \overline{x}_{11,11}) \\
\land (\overline{x}_{8,11} \lor \overline{x}_{10,11} \lor \overline{x}_{11,11})
\]
Some Examples of Hard Formulas w.r.t. Length (3/3)

**Subset cardinality formulas** [Spe10, VS10, MN14]

Variables = 1s in matrix with four 1s per row/column \( \text{+ extra } 1 \)

Row \( \Rightarrow \) majority of variables true; column \( \Rightarrow \) majority false

\[
\begin{pmatrix}
1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
\end{pmatrix}
\]

\[
(x_{1,1} \lor x_{1,2} \lor x_{1,4}) \\
\land (x_{1,1} \lor x_{1,2} \lor x_{1,8}) \\
\land (x_{1,1} \lor x_{1,4} \lor x_{1,8}) \\
\land (x_{1,2} \lor x_{1,4} \lor x_{1,8}) \\
\vdots \\
\land (\overline{x}_{4,11} \lor \overline{x}_{8,11} \lor \overline{x}_{10,11}) \\
\land (\overline{x}_{4,11} \lor \overline{x}_{8,11} \lor \overline{x}_{11,11}) \\
\land (\overline{x}_{4,11} \lor \overline{x}_{10,11} \lor \overline{x}_{11,11}) \\
\land (\overline{x}_{8,11} \lor \overline{x}_{10,11} \lor \overline{x}_{11,11})
\]
Some Examples of Hard Formulas w.r.t. Length (3/3)

**Subset cardinality formulas** [Spe10, VS10, MN14]

Variables = 1s in matrix with four 1s per row/column + extra 1

Row ⇒ majority of variables true; column ⇒ majority false

\[
\begin{pmatrix}
1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0
\end{pmatrix}
\]

\[
(x_{1,1} \lor x_{1,2} \lor x_{1,4})
\]

\[
\land (x_{1,1} \lor x_{1,2} \lor x_{1,8})
\]

\[
\land (x_{1,1} \lor x_{1,4} \lor x_{1,8})
\]

\[
\land (x_{1,2} \lor x_{1,4} \lor x_{1,8})
\]

\[
\vdots
\]

\[
\land (\overline{x}_{4,11} \lor \overline{x}_{8,11} \lor \overline{x}_{10,11})
\]

\[
\land (\overline{x}_{4,11} \lor \overline{x}_{8,11} \lor \overline{x}_{11,11})
\]

\[
\land (\overline{x}_{4,11} \lor \overline{x}_{10,11} \lor \overline{x}_{11,11})
\]

\[
\land (\overline{x}_{8,11} \lor \overline{x}_{10,11} \lor \overline{x}_{11,11})
\]
Some Examples of Hard Formulas w.r.t. Length (3/3)

**Subset cardinality formulas** [Spe10, VS10, MN14]

Variables = 1s in matrix with four 1s per row/column \(\oplus\) **extra 1**
Row \(\Rightarrow\) majority of variables true; column \(\Rightarrow\) majority false

\[
\begin{pmatrix}
1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
\end{pmatrix}
\]

\((x_{1,1} \lor x_{1,2} \lor x_{1,4}) \land (x_{1,1} \lor x_{1,2} \lor x_{1,8}) \land (x_{1,1} \lor x_{1,4} \lor x_{1,8}) \land (x_{1,2} \lor x_{1,4} \lor x_{1,8}) \land (x_{4,11} \lor x_{8,11} \lor x_{10,11}) \land (x_{4,11} \lor x_{8,11} \lor x_{11,11}) \land (x_{4,11} \lor x_{10,11} \lor x_{11,11}) \land (x_{8,11} \lor x_{10,11} \lor x_{11,11})\)

Lower bound \(\exp(\Omega(N))\) on expanding matrices (well spread-out)
Resolution Space

**Space** = \( \text{max } \# \text{ clauses in memory when performing refutation} \)

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

\[
\text{Space at step } t = \# \text{ clauses at steps } \leq t \text{ used at steps } \geq t
\]
**Resolution Space**

\[
\textbf{Space} = \text{max } \# \text{ clauses in memory} \quad \text{when performing refutation}
\]

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 \( t = \# \text{ clauses at steps } \leq t \text{ used at steps } \geq t \)

**Example:** Space at step 7 . . .

<table>
<thead>
<tr>
<th>Step</th>
<th>Clause</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( x \lor y )</td>
<td>Axiom</td>
</tr>
<tr>
<td>2</td>
<td>( x \lor \overline{y} \lor z )</td>
<td>Axiom</td>
</tr>
<tr>
<td>3</td>
<td>( \overline{x} \lor z )</td>
<td>Axiom</td>
</tr>
<tr>
<td>4</td>
<td>( \overline{y} \lor \overline{z} )</td>
<td>Axiom</td>
</tr>
<tr>
<td>5</td>
<td>( \overline{x} \lor \overline{z} )</td>
<td>Axiom</td>
</tr>
<tr>
<td>6</td>
<td>( x \lor \overline{y} )</td>
<td>Res(2, 4)</td>
</tr>
<tr>
<td>7</td>
<td>( x )</td>
<td>Res(1, 6)</td>
</tr>
<tr>
<td>8</td>
<td>( \overline{x} )</td>
<td>Res(3, 5)</td>
</tr>
<tr>
<td>9</td>
<td>( \bot )</td>
<td>Res(7, 8)</td>
</tr>
</tbody>
</table>
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 intrinsic 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 $N + O(1)$ (!) [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 $N + O(1)$ (!) [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 obtained for proofs of exponential size
Exist space-efficient proofs $\Rightarrow$ exist short proofs [AD08]
(for $k$-CNF formulas, to be precise)
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!
Exist **space-efficient proofs** ⇒ exist **short proofs** [AD08]
(for $k$-CNF formulas, to be precise)

Existence of short proofs ⇒ existence of space-efficient proofs? **No!**

**Pebbling formulas** [Nor09, NH13, BN08]
- Can be refuted in **length** $O(N)$
- May require **space** $\Omega(N/\log N)$
Pebbling Formulas

Encode so-called pebble games on DAGs [BW01]

1. \( u_1 \oplus u_2 \)
2. \( v_1 \oplus v_2 \)
3. \( w_1 \oplus w_2 \)
4. \((u_1 \oplus u_2) \land (v_1 \oplus v_2) \rightarrow (x_1 \oplus x_2)\)
5. \((v_1 \oplus v_2) \land (w_1 \oplus w_2) \rightarrow (y_1 \oplus y_2)\)
6. \((x_1 \oplus x_2) \land (y_1 \oplus y_2) \rightarrow (z_1 \oplus z_2)\)
7. \(\neg(z_1 \oplus z_2)\)

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

Encode so-called pebble games on DAGs [BW01]

1. \( u_1 \oplus u_2 \)
2. \( v_1 \oplus v_2 \)
3. \( w_1 \oplus w_2 \)
4. \( (u_1 \oplus u_2) \land (v_1 \oplus v_2) \rightarrow (x_1 \oplus x_2) \)
5. \( (v_1 \oplus v_2) \land (w_1 \oplus w_2) \rightarrow (y_1 \oplus y_2) \)
6. \( (x_1 \oplus x_2) \land (y_1 \oplus y_2) \rightarrow (z_1 \oplus z_2) \)
7. \( \neg (z_1 \oplus z_2) \)

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

Encode so-called **pebble games on DAGs** [BW01]

1. \( u_1 \oplus u_2 \)
2. \( v_1 \oplus v_2 \)
3. \( w_1 \oplus w_2 \)
4. \((u_1 \oplus u_2) \land (v_1 \oplus v_2) \rightarrow (x_1 \oplus x_2)\)
5. \((v_1 \oplus v_2) \land (w_1 \oplus w_2) \rightarrow (y_1 \oplus y_2)\)
6. \((x_1 \oplus x_2) \land (y_1 \oplus y_2) \rightarrow (z_1 \oplus z_2)\)
7. \(\neg(z_1 \oplus z_2)\)

- **sources** are true
- **truth** propagates upwards
- but **sink** is false
Pebbling Formulas

Encode so-called pebble games on DAGs [BW01]

1. \( u_1 \oplus u_2 \)
2. \( v_1 \oplus v_2 \)
3. \( w_1 \oplus w_2 \)
4. \( (u_1 \oplus u_2) \land (v_1 \oplus v_2) \rightarrow (x_1 \oplus x_2) \)
5. \( (v_1 \oplus v_2) \land (w_1 \oplus w_2) \rightarrow (y_1 \oplus y_2) \)
6. \( (x_1 \oplus x_2) \land (y_1 \oplus y_2) \rightarrow (z_1 \oplus z_2) \)
7. \( \neg(z_1 \oplus z_2) \)

- sources are true
- truth propagates upwards
- but sink is false
### Pebbling Formulas

Encode so-called **pebble games on DAGs** [BW01]

1. \( u_1 \oplus u_2 \)
2. \( v_1 \oplus v_2 \)
3. \( w_1 \oplus w_2 \)
4. \((u_1 \oplus u_2) \land (v_1 \oplus v_2) \rightarrow (x_1 \oplus x_2)\)
5. \((v_1 \oplus v_2) \land (w_1 \oplus w_2) \rightarrow (y_1 \oplus y_2)\)
6. \((x_1 \oplus x_2) \land (y_1 \oplus y_2) \rightarrow (z_1 \oplus z_2)\)
7. \(\neg(z_1 \oplus z_2)\)

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

Encode so-called pebble games on DAGs [BW01]

1. \( u_1 \oplus u_2 \)
2. \( v_1 \oplus v_2 \)
3. \( w_1 \oplus w_2 \)
4. \( (u_1 \oplus u_2) \land (v_1 \oplus v_2) \rightarrow (x_1 \oplus x_2) \)
5. \( (v_1 \oplus v_2) \land (w_1 \oplus w_2) \rightarrow (y_1 \oplus y_2) \)
6. \( (x_1 \oplus x_2) \land (y_1 \oplus y_2) \rightarrow (z_1 \oplus z_2) \)
7. \( \neg (z_1 \oplus z_2) \)

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

Encode so-called pebble games on DAGs [BW01]

1. \( u_1 \oplus u_2 \)
2. \( v_1 \oplus v_2 \)
3. \( w_1 \oplus w_2 \)
4. \((u_1 \oplus u_2) \land (v_1 \oplus v_2) \rightarrow (x_1 \oplus x_2)\)
5. \((v_1 \oplus v_2) \land (w_1 \oplus w_2) \rightarrow (y_1 \oplus y_2)\)
6. \((x_1 \oplus x_2) \land (y_1 \oplus y_2) \rightarrow (z_1 \oplus z_2)\)
7. \(\neg(z_1 \oplus z_2)\)

Write in CNF
E.g., \((x_1 \oplus x_2) \rightarrow (y_1 \oplus y_2)\) becomes

\[
(x_1 \lor \overline{x}_2 \lor y_1 \lor y_2) \land (x_1 \lor \overline{x}_2 \lor \overline{y}_1 \lor \overline{y}_2) \\
\land (\overline{x}_1 \lor x_2 \lor y_1 \lor y_2) \land (\overline{x}_1 \lor x_2 \lor \overline{y}_1 \lor \overline{y}_2)
\]
Pebbling Formulas

Encode so-called pebble games on DAGs [BW01]

1. \( u_1 \oplus u_2 \)
2. \( v_1 \oplus v_2 \)
3. \( w_1 \oplus w_2 \)
4. \((u_1 \oplus u_2) \land (v_1 \oplus v_2) \rightarrow (x_1 \oplus x_2)\)
5. \((v_1 \oplus v_2) \land (w_1 \oplus w_2) \rightarrow (y_1 \oplus y_2)\)
6. \((x_1 \oplus x_2) \land (y_1 \oplus y_2) \rightarrow (z_1 \oplus z_2)\)
7. \( \neg (z_1 \oplus z_2) \)

Write in CNF

E.g., \((x_1 \oplus x_2) \rightarrow (y_1 \oplus y_2)\) becomes

\[
(x_1 \lor \overline{x}_2 \lor y_1 \lor y_2) \land (x_1 \lor \overline{x}_2 \lor \overline{y}_1 \lor \overline{y}_2) \\
\land (\overline{x}_1 \lor x_2 \lor y_1 \lor y_2) \land (\overline{x}_1 \lor x_2 \lor \overline{y}_1 \lor \overline{y}_2)
\]

Pebbling space lower bounds \(\Rightarrow\) resolution space lower bounds
Pebbling Formulas

Encode so-called pebble games on DAGs [BW01]

1. $u_1 \oplus u_2$
2. $v_1 \oplus v_2$
3. $w_1 \oplus w_2$
4. $(u_1 \oplus u_2) \land (v_1 \oplus v_2) \rightarrow (x_1 \oplus x_2)$
5. $(v_1 \oplus v_2) \land (w_1 \oplus w_2) \rightarrow (y_1 \oplus y_2)$
6. $(x_1 \oplus x_2) \land (y_1 \oplus y_2) \rightarrow (z_1 \oplus z_2)$
7. $\neg (z_1 \oplus z_2)$

Write in CNF
E.g., $(x_1 \oplus x_2) \rightarrow (y_1 \oplus y_2)$ becomes

$$(x_1 \lor \overline{x_2} \lor y_1 \lor y_2) \land (x_1 \lor \overline{x_2} \lor \overline{y_1} \lor \overline{y_2})$$
$$\land (\overline{x_1} \lor x_2 \lor y_1 \lor y_2) \land (\overline{x_1} \lor x_2 \lor \overline{y_1} \lor \overline{y_2})$$

Pebbling space lower bounds $\Rightarrow$ resolution space lower bounds
(Works also for other functions than $\oplus$)
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 $\approx$ running time; space $\approx$ 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 $x_i \overset{d}{=} b$ and unit propagations $x_i \overset{C}{=} b$

\[
\begin{align*}
(x_7 \overset{d}{=} 0, & \quad x_2 \overset{d}{=} 1, x_{12} \overset{C_1}{=} 0, x_6 \overset{d}{=} 1, x_4 \overset{C_2}{=} 1, x_1 \overset{C_3}{=} 0, x_{11} \overset{d}{=} 0, x_{59} \overset{C_4}{=} 1) \\
\text{dec. level 1} & \quad \text{decision level 2} & \quad \text{decision level 3} & \quad \text{decision level 4}
\end{align*}
\]
Abstract Description of CDCL (1/2)

**Trail:** a stack of decisions $x_i^d = b$ and unit propagations $x_i^C = b$

\[
\begin{aligned}
(x_7 &\overset{d}{=} 0, x_2 \overset{d}{=} 1, x_{12} \overset{C_1}{=} 0, x_6 \overset{d}{=} 1, x_4 \overset{C_2}{=} 1, x_1 \overset{C_3}{=} 0, x_{11} \overset{d}{=} 0, x_{59} \overset{C_4}{=} 1) \\
\text{dec. level 1} &\quad \text{decision level 2} &\quad \text{decision level 3} &\quad \text{decision level 4}
\end{aligned}
\]

**Clause database $\mathcal{D}$:** original formula $+$ learned clauses
Abstract Description of CDCL (1/2)

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

( $x_7 \overset{d}{=} 0$, $x_2 \overset{d}{=} 1$, $x_{12} \overset{C_1}{=} 0$, $x_6 \overset{d}{=} 1$, $x_4 \overset{C_2}{=} 1$, $x_1 \overset{C_3}{=} 0$, $x_{11} \overset{d}{=} 0$, $x_{59} \overset{C_4}{=} 1$ )

decision level 1 \hspace{1cm} \text{decision level 2} \hspace{1cm} \text{decision level 3} \hspace{1cm} \text{decision level 4}

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

Start in **Default** mode; transit to **Conflict**, **Unit**, or **Decision**
Abstract Description of CDCL (1/2)

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

\[
\begin{align*}
(x_7 \overset{d}{=} 0, x_2 \overset{d}{=} 1, x_{12} \overset{C_1}{=} 0, x_6 \overset{d}{=} 1, x_4 \overset{C_2}{=} 1, x_1 \overset{C_3}{=} 0, x_{11} \overset{d}{=} 0, x_{59} \overset{C_4}{=} 1)
\end{align*}
\]

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

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

**Default** If trail falsifies clause $C \in \mathcal{D}$, move to **Conflict**;
Abstract Description of CDCL (1/2)

**Trail:** a stack of decisions \( x_i \overset{d}{=} b \) and unit propagations \( x_i \overset{C}{=} b \)

\[
\begin{align*}
&x_7 \overset{d}{=} 0, x_2 \overset{d}{=} 1, x_{12} \overset{C_1}{=} 0, x_6 \overset{d}{=} 1, x_4 \overset{C_2}{=} 1, x_1 \overset{C_3}{=} 0, x_{11} \overset{d}{=} 0, x_{59} \overset{C_4}{=} 1 \\
\text{dec. level 1} & \quad \text{decision level 2} & \quad \text{decision level 3} & \quad \text{decision level 4}
\end{align*}
\]

**Clause database** \( \mathcal{D} \): original formula + learned clauses

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

**Default** 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 \( x_i \overset{d}{=} b \) and unit propagations \( x_i \overset{C}{=} b \)

\[
\begin{align*}
&x_7 \overset{d}{=} 0, \quad x_2 \overset{d}{=} 1, \quad x_{12} \overset{C_1}{=} 0, \quad x_6 \overset{d}{=} 1, \quad x_4 \overset{C_2}{=} 1, \quad x_1 \overset{C_3}{=} 0, \quad x_{11} \overset{d}{=} 0, \quad x_{59} \overset{C_4}{=} 1 \\
&\text{dec. level 1} \quad \text{decision level 2} \quad \text{decision level 3} \quad \text{decision level 4}
\end{align*}
\]

**Clause database** \( \mathcal{D} \): original formula + learned clauses

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

- **Default**  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 $x_i \overset{d}{=} b$ and unit propagations $x_i \overset{C}{=} b$

( $x_7 \overset{d}{=} 0$, $x_2 \overset{d}{=} 1$, $x_{12} \overset{C_1}{=} 0$, $x_6 \overset{d}{=} 1$, $x_4 \overset{C_2}{=} 1$, $x_1 \overset{C_3}{=} 0$, $x_{11} \overset{d}{=} 0$, $x_{59} \overset{C_4}{=} 1$ )

dec. level 1 decision level 2 decision level 3 decision level 4

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

Start in Default mode; transit to Conflict, Unit, or Decision

Default 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 Default;
Abstract Description of CDCL (1/2)

**Trail:** a stack of decisions $x_i^d = b$ and unit propagations $x_i^C = b$

$\left( \begin{array}{c}
    x_7^d = 0 \\
    x_2^d = 1 \\
    x_{12}^C = 0 \\
    x_6^d = 1 \\
    x_4^C = 1 \\
    x_1^C = 0 \\
    x_{11}^d = 0 \\
    x_{59}^C = 1
\end{array} \right)$

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

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

**Default** 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 **Default**;
else
1. decide if to apply database reduction to $\mathcal{D}$;
2. move to **Decision**
Abstract Description of CDCL (2/2)

**Unit** Pick clause $C \in D$ that is unit w.r.t. trail
Add propagated assignment $x_C = b$ to trail
Move to Default
Abstract Description of CDCL (2/2)

**Unit**  Pick clause $C \in D$ that is unit w.r.t. trail
   Add propagated assignment $x^C = b$ to trail
   Move to Default

**Conflict**  If trail contains no decisions, output UNSAT;
   else
      • apply learning scheme to derive asserting clause $C'$;
      • backjump, i.e., remove decision levels $>\$ assertion level of $C'$ from trail;
      • move to **Unit**
Abstract Description of CDCL (2/2)

**Unit**  
Pick clause $C \in D$ that is unit w.r.t. trail  
Add propagated assignment $x \overset{C}{=} b$ to trail  
Move to **Default**

**Conflict**  
If trail contains no decisions, output UNSAT;  
else  
  • apply **learning scheme** to derive asserting clause $C'$;  
  • backjump, i.e., remove decision levels $> \text{assertion level of } C'$ from trail;  
  • move to **Unit**

**Decision**  
Use **decision scheme** to add decision $x \overset{d}{=} b$ to trail  
Move to **Default**
Abstract Description of CDCL (2/2)

**Unit**  Pick clause $C \in D$ that is unit w.r.t. trail
            Add propagated assignment $x_C = b$ to trail
            Move to **Default**

**Conflict** If trail contains no decisions, output UNSAT;  
             else  
             • apply **learning scheme** to derive asserting  
                clause $C'$;  
             • backjump, i.e., remove decision levels $>  
                assertion level of $C'$ from trail;
             • move to **Unit**

**Decision** Use **decision scheme** to add decision $x_d = b$ to trail
              Move to **Default**

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

Too small formula for interesting example... 

\[(x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z})\]
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:

\[(u \lor x \lor y) \land (x \lor \bar{y} \lor z) \land (\bar{x} \lor z) \land (\bar{y} \lor \bar{z}) \land (\bar{x} \lor \bar{z}) \land (u \lor w) \land (\bar{u} \lor \bar{w})\]
CDCL Execution Example

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

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

\[
w \overset{d}{=} 0
\]
CDCL Execution Example

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

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

\(d\)

\[w \models 0\]

\[u \models 0\]
CDCL Execution Example

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

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

\[
\begin{array}{c}
w \doteq 0 \\
\overline{u} \lor w \\
x \doteq 0
\end{array}
\]
CDCL Execution Example

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

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

\[
\begin{align*}
\ & w^d = 0 \\
\ & u^d = 0 \\
\ & x^d = 0 \\
\ & u^d \lor x^d \lor y^d = 1
\end{align*}
\]
CDCL Execution Example

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

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

\[
\begin{array}{c}
w \overset{d}{=} 0 \\
u \overset{0}{=} 0 \\
x \overset{0}{=} 0 \\
y \overset{1}{=} 1 \\
z \overset{1}{=} 1
\end{array}
\]
CDCL Execution Example

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

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

\[
\begin{align*}
w & \equiv 0 \\
u & \equiv 0 \\
x & \equiv 0 \\
y & \equiv 1 \\
z & \equiv 1 \\
\overline{y} \lor \overline{z} & \equiv \bot
\end{align*}
\]
CDCL Execution Example

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

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

\[
\begin{align*}
  w &\equiv 0 \\
  u &\equiv 0 \\
  x &\equiv 0 \\
  y &\equiv y_1 = 1 \\
  z &\equiv z_1 = 1 \\
  \overline{y} \lor \overline{z} &\equiv \bot
\end{align*}
\]
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:

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

\[
\begin{align*}
  w &\equiv 0 \\
  u &\equiv 0 \\
  x &\equiv 0 \\
  y &\equiv 1 \\
  z &\equiv 1 \\
  \overline{y} \lor \overline{z} &\equiv 1
\end{align*}
\]
CDCL Execution Example

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

\[(u \lor x \lor y) \land (x \lor \bar{y} \lor z) \land (\bar{x} \lor z) \land (\bar{y} \lor \bar{z}) \land (\bar{x} \lor \bar{z}) \land (\bar{u} \lor w) \land (\bar{u} \lor \bar{w})\]

\[
\begin{align*}
&w \overset{d}{=} 0 \\
&u \overset{0}{=} 0 \\
&x \overset{0}{=} 0 \\
&u \lor x \overset{1}{=} \\
&x \lor \bar{y} \overset{1}{=} \\
&z \lor \bar{y} \overset{1}{=} \\
&z \overset{1}{=} \\
&y \lor \bar{z} \\
&\perp
\end{align*}
\]
CDCL Execution Example

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

\[(u \lor x \lor y) \land (x \lor y \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor z) \land (\overline{x} \lor \overline{z}) \land (\overline{u} \lor w) \land (\overline{u} \lor \overline{w})\]
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:

\[(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (u \lor w) \land (u \lor \overline{w})\]
CDCL Execution Example

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

\[(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor z) \land (\overline{x} \lor \overline{z}) \land (u \lor w) \land (\overline{u} \lor \overline{w})\]
CDCL Execution Example

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

\[(u \lor x \lor y) \land (x \lor \neg y \lor z) \land (\neg x \lor z) \land (\neg y \lor \neg z) \land (\neg \neg x \lor \neg z) \land (\neg u \lor w) \land (\neg \neg u \lor \neg w)\]

\[
\begin{align*}
& w = 0 \\
& u = 0 \\
& x = 0 \\
& y = 1 \\
& z = 1 \\
& \neg \neg x = 1 \\
& \neg \neg \neg x = 1 \\
& \neg \neg \neg \neg x = 1 \\
\end{align*}
\]
CDCL Execution Example

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

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

```
\begin{align*}
  w & = 0 \\
  u & \lor w = 0 \\
  x & = 0 \\
  y & \lor x \lor y = 1 \\
  z & \lor \overline{y} \lor z = 1 \\
  \overline{y} \lor \overline{z} & = 1 \\
\end{align*}
```

```
\begin{align*}
  w & = 0 \\
  u & \lor w = 0 \\
  x & = 1 \\
  u \lor x & = 1 \\
  w & = 1 \\
  u \lor w & = 1 \\
\end{align*}
```

```
\begin{align*}
  u \lor x & \\
  x \lor \overline{y} & \\
  x \lor \overline{y} \lor z & = 1 \\
  \overline{x} \lor \overline{z} & = 1 \\
\end{align*}
```

\(x \lor \overline{y} \lor z = 1\)
CDCL Execution Example

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

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

\[
\begin{array}{c}
w_d = 0 \\
u \lor w = 0 \\
x_d = 0 \\
u \lor x \lor y = 1 \\
x \lor \overline{y} = 1 \\
x \lor \overline{y} \lor z = 1 \\
x \lor \overline{y} \lor z = 1 \\
\overline{y} \lor \overline{z} = 1
\end{array}
\]

\[
\begin{array}{c}
w_d = 0 \\
u \lor w = 0 \\
x = 0 \\
u \lor x = 1 \\
\overline{u} \lor w = 1 \\
\overline{u} \lor \overline{w} = 1 \\
\overline{u} = 0
\end{array}
\]

Jakob Nordström (KTH) Understanding CDCL Through Lens of Proof Complexity Fields Institute Aug ’16 22/51
CDCL Execution Example

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

\[(u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (u \lor w) \land (\overline{u} \lor \overline{w})\]
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...

\[
\begin{align*}
\frac{u \lor v}{w} &= 0 \\
\frac{u \lor w}{u} &= 0 \\
\frac{x}{d} &= 0 \\
\frac{y \lor x \lor y}{z} &= 1 \\
\frac{x \lor \overline{y}}{z} &= 1 \\
\frac{\overline{y} \lor \overline{z}}{\bot} &= 1 \\
\frac{w}{d} &= 0 \\
\frac{u \lor v}{u} &= 0 \\
\frac{x}{u \lor x}{1} &= 1 \\
\frac{x \lor \overline{z}}{z} &= 1 \\
\frac{\overline{u} \lor \overline{w}}{\bot} &= 1 \\
\end{align*}
\]
CDCL Execution Example as Resolution Refutation

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

\[
\begin{align*}
    w^d & = 0 \\
    u & = 0 \\
    x^d & = 0 \\
    u \lor \neg w & \\
    u \lor \neg w & \\
    u \lor \neg w & \\
    \neg u \lor \neg w & \\
    u \lor \neg x & \\
    \neg u \lor w & \\
    \neg u \lor \neg w & \\
    \neg x \lor z & \\
    x \lor \neg y \lor z & \\
    \neg y \lor z & \\
\end{align*}
\]
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 \(\Rightarrow\) 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 $\Rightarrow$ 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(!)
Proof Plan for 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 \( t = 1, 2, 3, \ldots \)
Proof Plan for 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...
- Unit propagation + clause database cause problems
- Suppose have \(C \lor x\) and \(D \lor \bar{x}\) and now want to learn \(C \lor D\)
- Why not just decide to make \(C \lor D\) false \(\Rightarrow\) conflict on \(x\)?!
- Might not be possible: other clauses can propagate literals to “wrong values” \(\Rightarrow\) proof search veers off in different direction
- And even if possible, might not learn \(C \lor D\)
Proof Plan for 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...

- Unit propagation + clause database cause problems
- Suppose have \(C \lor x\) and \(D \lor \overline{x}\) and now want to learn \(C \lor D\)
- Why not just decide to make \(C \lor D\) false \(\Rightarrow\) conflict on \(x\)?!
- Might not be possible: other clauses can propagate literals to “wrong values” \(\Rightarrow\) proof search veers off in different direction
- And even if possible, might not learn \(C \lor D\)

Non-standard assumptions needed precisely for these reasons
CDCL Simulation of Resolution

- First result in clean model in [PD11]: **CDCL as proof system polynomially simulates resolution w.r.t. time/size**
CDCL Simulation of Resolution

- First result in clean model in [PD11]: CDCL as proof system polynomially simulates resolution w.r.t. time/size
- Constructive version in [AFT11]: \exists \text{ resolution proof with clauses of bounded size } \Rightarrow \text{ CDCL will run fast}
CDCL Simulation of Resolution

- 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.
  Can use techniques in either paper to establish results in other.
CDCL Simulation of Resolution

- First result in clean model in [PD11]: CDCL as proof system polynomially simulates resolution w.r.t. time/size

- Constructive version in [AFT11]: ∃ resolution proof with clauses of bounded size ⇒ CDCL will run fast

- [AFT11] and [PD11] independent but essentially equivalent
  Can use techniques in either paper to establish results in other

- **Key insight:** Don’t have to learn *exactly* clauses $C_t$ in proof
CDCL Simulation of Resolution

- 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

- [AFT11] and [PD11] independent but essentially equivalent
  Can use techniques in either paper to establish results in other

- Key insight: Don’t have to learn exactly clauses \( C_t \) in proof

- Enough to learn other clauses yielding at least same unit propagations as \( C_t \) (absorption)
CDCL Simulation of Resolution

- 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
- [AFT11] and [PD11] independent but essentially equivalent
  Can use techniques in either paper to establish results in other
- Key insight: Don’t have to learn exactly clauses \( C_t \) in proof
- Enough to learn other clauses yielding at least same unit propagations as \( C_t \) (absorption)
- Good, so then we’re done understanding CDCL? Not quite...
Room for Further Improvement of [AFT11, PD11]? (1/2)

**Learning scheme**
- Learned clause assertive but otherwise adversarially chosen
- Very strong aspect of result
- But does not come for free — costs a lot for efficiency of simulation
Room for Further Improvement of [AFT11, PD11]? (1/2)

Learning scheme
- Learned clause assertive but otherwise adversarially chosen
- Very strong aspect of result
- But does not come for free — costs a lot for efficiency of simulation

Restart policy
- Restarts are *not too frequent* (unless Luby is too frequent)
- But no progress at all in between restarts
- Restarts seem important, but not quite *that* important?!
Room for Further Improvement of [AFT11, PD11]? (2/2)

Decision strategy

- In [PD11], crucially relies on (unknown) resolution proof
- In [AFT11], crucially needs to be (essentially totally) random
- Probably inherent — fully constructive proof search likely to be computationally intractable [AR08]
Decision strategy

- In [PD11], crucially relies on (unknown) resolution proof
- In [AFT11], crucially needs to be (essentially totally) random
- Probably inherent — fully constructive proof search likely to be computationally intractable [AR08]

Clause database management

- No learned clause must ever be forgotten, or theorems crash and burn
- But in practice something like 90–95% of clauses erased...
Room for Further Improvement of [AFT11, PD11]? (2/2)

**Decision strategy**
- In [PD11], crucially relies on (unknown) resolution proof
- In [AFT11], crucially needs to be (essentially totally) random
- Probably inherent — fully constructive proof search likely to be computationally intractable [AR08]

**Clause database management**
- No learned clause must ever be forgotten, or theorems crash and burn
- But in practice something like 90–95% of clauses erased...

**Simulation efficiency**
- Solvers typically have to run in (close to) linear time $O(n)$
- But simulation will yield something like $O(n^5)$ running time
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

**Proof verification:** check execution trace for
- Full and correct unit propagation
- Decisions only when no possible propagation or conflict
- Clauses learned in accordance with learning scheme
- No erasures of active reason clauses on trail
- Et cetera... (see paper for details)
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
- **Proof verification:** check execution trace for
  - Full and correct unit propagation
  - Decisions only when no possible propagation or conflict
  - Clauses learned in accordance with learning scheme
  - No erasures of active reason clauses on trail
  - Et cetera... (see paper for details)

- **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 $\Rightarrow$ collapse to tree-like resolution
- Show too aggressive clause removal $\Rightarrow$ exponential blow-up in running time, matching theory [BN11, BBI12, BNT13]
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)
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])
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 $\Rightarrow$ 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
- Cannot locally verify proof (doubleplusunnice)
Sanity Check: CDCL Cannot Do Better than Resolution

Theorem ([EJL+16])

If CDCL with “standard” learning scheme (e.g., 1UIP) decides $F$ in time $\tau$ and space $s$
then $F$ has resolution proof in size $\leq \tau$ and space $\leq s + O(1)$
Sanity Check: CDCL Cannot Do Better than Resolution

Theorem ([EJL+16])

If CDCL with “standard” learning scheme (e.g., 1UIP) decides $F$ in time $\tau$ and space $s$ then $F$ has resolution proof in size $\leq \tau$ and space $\leq s + O(1)$

Fairly obvious for time/size
Sanity Check: CDCL Cannot Do Better than Resolution

Theorem ([EJL+16])

If CDCL with “standard” learning scheme (e.g., 1UIP) decides $F$ in time $\tau$ and space $s$ then $F$ has resolution proof in size $\leq \tau$ and space $\leq s + O(1)$

Fairly obvious for time/size

A priori not so obvious for space
(but proof not hard once one gets the model right)
Sanity Check: CDCL Cannot Do Better than Resolution

Theorem ([EJL⁺16])

**If** CDCL with “standard” learning scheme (e.g., 1UIP) decides $F$

in time $\tau$ and space $s$

**then** $F$ has resolution proof in size $\leq \tau$ and space $\leq s + O(1)$

Fairly obvious for time/size

A priori not so obvious for space

(but proof not hard once one gets the model right)

So lower bounds in resolution trade-offs automatically carry over

But can CDCL find time-efficient and space-efficient proofs?
Time-Space Trade-Offs for CDCL (in Math Notation)

We obtain CDCL analogues of (almost all) trade-off results in [BN11, BBI12, BNT13] — here is one sample:

**Theorem ([EJL+16], slightly informal)**

For your favourite $k \in \mathbb{N}^+$ there exist explicit formulas $F_N$ of size $\approx N$ such that

- CDCL with 1UIP learning and no restarts can decide $F_N$ in time $O(N^k)$ and space $O(N^k)$
- CDCL with 1UIP learning and no restarts can decide $F_N$ in space $O(\log^2 N)$ and time $N^{O(\log N)}$
- For any CDCL run in time $\tau$ and space $s$ using any learning scheme and restart policy it holds that $\tau \gtrapprox (N^k/s)^\Omega(\log \log N / \log \log \log N)$
Time-Space Trade-Offs for CDCL (in English)

Very informal statement of theorem to convey high-level message:

- Somewhat tricky formulas $F_N$ (require superlinear time)
- CDCL can solve them efficiently for generous memory management (even without restarts)
- But more aggressive clause erasure policy (such as current MiniSat or Glucose defaults) cause superpolynomial blow-up in running time
Time-Space Trade-Offs for CDCL (in English)

Very informal statement of theorem to convey high-level message:

- Somewhat tricky formulas $F_N$ (require superlinear time)
- CDCL can solve them efficiently for generous memory management (even without restarts)
- But more aggressive clause erasure policy (such as current MiniSat or Glucose defaults) cause superpolynomial blow-up in running time

Interpretation:

- This is only a mathematical theorem about asymptotic properties of theoretical benchmarks
- But have some indications of similar behaviour for scaled-down versions 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:** $x \lor y \lor \overline{z}$ 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 \(1 \equiv \text{true}\) and \(0 \equiv \text{false}\))

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

**Derivation rules**

<table>
<thead>
<tr>
<th>Variable axioms</th>
<th>Multiplication</th>
<th>Division</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0 \leq x \leq 1)</td>
<td>(\sum a_i x_i \geq A) (\sum c a_i x_i \geq c A)</td>
<td>(\sum c a_i x_i \geq A) (\sum a_i x_i \geq \lceil A/c \rceil)</td>
</tr>
</tbody>
</table>

**Addition** \(\sum a_i x_i \geq A\) \(\sum b_i x_i \geq B\) \(\sum (a_i + b_i) x_i \geq A + B\)

**Goal:** Derive \(0 \geq 1 \iff\) 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

$q_{k,1} \lor q_{k,2} \lor \cdots \lor q_{k,n}$
\quad some vertex is $k$th member of clique

$\overline{q}_{k,i} \lor \overline{q}_{k,j}$
\quad $k$th clique member is uniquely defined

$p_{i,j} \lor \overline{q}_{k,i} \lor \overline{q}_{k',j}$
\quad clique members are connected by edges

$r_{i,1} \lor r_{i,2} \lor \cdots \lor r_{i,m-1}$
\quad every vertex $i$ has a colour

$\overline{p}_{i,j} \lor \overline{r}_{i,\ell} \lor \overline{r}_{j,\ell}$
\quad neighbours have distinct colours

Exponential lower bound via interpolation and circuit complexity
Technique very specifically tied to structure of formula
Open Problems for Cutting Planes Length and Space

**Open Problems**

Prove *length lower bounds* for cutting planes

- for *Tseitin formulas*
- for *random* $k$-*CNFs*
- for any formula using *other technique than interpolation*
Open Problems for Cutting Planes Length and Space

Open Problems

Prove length lower bounds for cutting planes
- for Tseitin formulas
- for random $k$-CNFs
- for any formula using other technique than interpolation

Open Problems

Prove space lower bounds for cutting planes
- with constant-size coefficients (very weak bounds in [GPT15])
- with polynomial-size coefficients (nothing known)
Size-Space Trade-offs for Cutting Planes?

- Short cutting planes refutations of (lifted) Tseitin formulas on expanders need large space [GP14] (but probably don’t exist)
- Short cutting planes refutations of (some) pebbling formulas need large space [HN12, GP14] (and such refutations exist)

Results obtained via communication complexity
Size-Space Trade-offs for Cutting Planes?

- Short cutting planes refutations of (lifted) Tseitin formulas on expanders need large space [GP14] (but probably don’t exist)
- Short cutting planes refutations of (some) pebbling formulas need large space [HN12, GP14] (and such refutations exist)

Results obtained via communication complexity

By [GPT15] get trade-offs with “constant space” upper bounds (but with coefficients of exponential size)

Doesn’t seem like a too relevant a trade-off — exponential size coefficients doesn’t feel like “small space”
Size-Space Trade-offs for Cutting Planes?

- Short cutting planes refutations of \((\text{lifted})\) Tseitin formulas on expanders need large space [GP14] (but probably don’t exist)
- Short cutting planes refutations of \((\text{some})\) pebbling formulas need large space [HN12, GP14] (and such refutations exist)

Results obtained via communication complexity

By [GPT15] get trade-offs with “constant space” upper bounds (but with coefficients of exponential size).

Doesn’t seem like a too relevant a trade-off — exponential size coefficients doesn’t feel like “small space”

Open Problem

Are there \textit{trade-offs where the space-efficient CP refutations have small coefficients?} (Say, of polynomial or even constant size)
Recent news: Yes, there are such trade-offs!

Theorem ([dRNV16])

*There exist flavours of pebbling formulas such that*

- ∃ small-size refutations with constant-size coefficients
- ∃ small-space refutations with constant-size coefficients
- Decreasing the space even for refutations with exponentially large coefficients causes exponential blow-up of length
Size-Space Trade-offs for Cutting Planes!

**Recent news:** Yes, there are such trade-offs!

**Theorem ([dRNV16])**

There exist flavours of pebbling formulas such that

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

- Again uses communication complexity (+ several other twists)
- Downside: Parameters worse than in previous results
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).
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
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?
Conflict-Driven CP Solvers: Two Concrete Obstacles

- **Roadblock 1:** Given CNF input, solvers cannot discover and use cardinality constraints (too limited form of addition)
Conflict-Driven CP Solvers: Two Concrete Obstacles

- **Roadblock 1:** Given CNF input, solvers cannot discover and use **cardinality constraints** *(too limited form of addition)*

- But given more helpful encoding, solvers can do really well *(e.g., PHP and subset cardinality formulas)* [BLLM14]
Conflict-Driven CP Solvers: Two Concrete Obstacles

- **Roadblock 1**: Given CNF input, solvers cannot discover and use cardinality constraints (too limited form of addition)

- But given more helpful encoding, solvers can do really well (e.g., PHP and subset cardinality formulas) [BLLM14]

- **Roadblock 2(?)**: Solvers seem inefficient for systems of inequalities that have rational but not integral solutions (too limited form of division?)
Conflict-Driven CP Solvers: Two Concrete Obstacles

- **Roadblock 1:** Given CNF input, solvers cannot discover and use cardinality constraints (too limited form of addition).

- But given more helpful encoding, solvers can do really well (e.g., PHP and subset cardinality formulas) [BLLM14]

- **Roadblock 2(?)** Solvers seem inefficient for systems of inequalities that have rational but not integral solutions (too limited form of division?)

- Fail on, e.g., even colouring formulas [Mar06] for no obvious good reason

- Not well understood at all — work in progress
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 (though might incur some blow-up)
  - ... given right variable decision order

Experimental Set-up
Some Tentative Findings
Theoretically Easy Combinatorical 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 (though might incur some blow-up)
  - ... 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 (though might incur some blow-up)
  - ... 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
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 (though might incur some blow-up)
  - . . . 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

- Practical note: many (though not quite all) formulas generated using the tool **CNFgen** [CNF, LENV16]
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 “conventional 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

Tseitin grid (5xN): different restart and clause erasure strategies

- Minisat 2.2 reduce freq, no restarts
- Minisat 2.2 reduce freq, LBD restarts
- LBD reduce freq, no restarts
- LBD reduce freq, LBD restarts
- No deletion, no restarts
- No deletion, LBD restarts
- Fixed var. order, no deletion
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**
- Phase saving only helps together with frequent restarts
- Sometimes small variations in VSIDS decay factor (rate of forgetting) 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
- Phase saving only helps together with frequent restarts
- Sometimes small variations in VSIDS decay factor (rate of forgetting) 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 (Tseitin, even colouring) — VSIDS just goes dead wrong
- Sometimes strange easy-hard-easy patterns (subset cardinality)
Plot 2: Ordering Principle Formulas

POP: different VSIDS decay factor and restart strategies

- VSIDS 0.95, No restarts
- VSIDS 0.95, LBD restarts
- VSIDS 0.80, No restarts
- VSIDS 0.80, LBD restarts
- Fixed var. order

Jakob Nordström (KTH)
Understanding CDCL Through Lens of Proof Complexity
Fields Institute Aug ’16 49/51
Plot 3: Subset Cardinality Formulas

Subset card: different clause erasure and restart strategies

- LBD assessment, Minisat 2.2 reduce freq, no restarts
- LBD assessment, Minisat 2.2 reduce freq, LBD restarts
- LBD assessment, LBD reduce freq, no restarts
- LBD assessment, LBD reduce freq, LBD restarts

Time (s)

N
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?
- Are there formulas for which VSIDS goes provably wrong?
- 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!

Jakob Nordström (KTH)
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?
- Are there formulas for which VSIDS goes provably wrong?
- 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?
- Are there formulas for which VSIDS goes provably wrong?
- 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


References II


References III


<table>
<thead>
<tr>
<th>Reference</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[CNF]</td>
<td>CNFgen formula generator and tools. <a href="https://github.com/MassimoLauria/cnfgen">link</a></td>
</tr>
</tbody>
</table>


http://www.labri.fr/perso/lsimon/glucose/.

### References

<table>
<thead>
<tr>
<th>Reference</th>
<th>Author(s)</th>
<th>Title</th>
</tr>
</thead>
</table>


References


References XI


<table>
<thead>
<tr>
<th>Reference</th>
<th>Author(s)</th>
<th>Title</th>
<th>Conference Details</th>
</tr>
</thead>
</table>