Trade-offs Between Time and Memory in a Tighter Model of CDCL SAT Solvers

Jakob Nordström

KTH Royal Institute of Technology
Stockholm, Sweden

19th International Conference on Theory and Applications of Satisfiability Testing
Bordeaux, France
July 5, 2016

Joint work with Jan Elffers, Jan Johannsen, Massimo Lauria, Thomas Magnard, and Marc Vinyals

What This Work Is About

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 the limits to what they can do?

What This Work Is About

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 the limits to what they can do?

This work

- Driving motivation: **Understand the power of CDCL**
- Tool: **Proof complexity** (don't have much else for rigorous analysis)

What This Talk Is About

- Report on results so far
- Definitely more of "work in progress" than The Final Answer$^{\text{TM}}$
- Also take the opportunity to give my take on some work at intersection of SAT solving and proof complexity
- Believe there is room for improved mutual understanding — hope to stimulate discussions that can remove some misconceptions

CDCL and Resolution

- Satisfiable CNF formula: CDCL solver finds satisfying assignment

CDCL and Resolution

- Satisfiable CNF formula: CDCL solver finds satisfying assignment
- Unsatisfiable formula: search for proof in resolution proof system

CDCL and Resolution

- Satisfiable CNF formula: CDCL solver finds satisfying assignment
- Unsatisfiable formula: search for proof in resolution proof system
- Lower bounds in proof complexity ⇒ impossibility results for CDCL even assuming optimal choices*

CDCL and Resolution

- Satisfiable CNF formula: CDCL solver finds satisfying assignment
- Unsatisfiable formula: search for proof in resolution proof system
- Lower bounds in proof complexity $\Rightarrow$ impossibility results for CDCL even assuming optimal choices*
- But CDCL searches for proofs with very special structure — can it match resolution upper bounds?

CDCL and Resolution

- Satisfiable CNF formula: CDCL solver finds satisfying assignment
- Unsatisfiable formula: search for proof in resolution proof system
- Lower bounds in proof complexity ⇒ impossibility results for CDCL even assuming optimal choices*
- But CDCL searches for proofs with very special structure — can it match resolution upper bounds?

(*) Ignores preprocessing — our focus on CDCL proof search
Will be happy to elaborate offline on why this is reasonable simplification

Understanding the Efficiency of CDCL Proof Search

- 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

- 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 strategies
  - no unit propagation(!)

Understanding the Efficiency of CDCL Proof Search

- 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 strategies
  - no unit propagation(!)
- First result in clean model in [PD11]: CDCL as proof system polynomially simulates resolution w.r.t. time/size

Understanding the Efficiency of CDCL Proof Search

- 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 strategies
  - no unit propagation(!)
- 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

Understanding the Efficiency of CDCL Proof Search

- 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 strategies
  - no unit propagation(!)
- First result in clean model in [PD11]: CDCL as proof system polynomially simulates resolution w.r.t. time/size
- Constructive version in [AFT11]:* $\exists$ resolution proof with clauses of bounded size $\Rightarrow$ CDCL will run fast
- Good, so then we're done? Not quite

Understanding the Efficiency of CDCL Proof Search

- 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 strategies
  - no unit propagation(!)
- First result in clean model in [PD11]: CDCL as proof system polynomially simulates resolution w.r.t. time/size
- Constructive version in [AFT11]:* $\exists$ resolution proof with clauses of bounded size $\Rightarrow$ CDCL will run fast
- Good, so then we're done? Not quite

(*) [AFT11] and [PD11] independent but essentially equivalent works
Can use techniques in either paper to establish results in the other

Why Not Completely Happy with [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

Why Not Completely Happy with [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 you think Luby is too frequent)
- But no progress at all in between restarts
- Restarts seem important, but not quite *that* important?!

Why Not Completely Happy with [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]

Why Not Completely Happy with [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...

Why Not Completely Happy with [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

- CDCL solvers typically have to run in (close to) linear time $\mathcal{O}(n)$
- But simulation will yield something like $\mathcal{O}(n^5)$ running time

What We Want

More fine-grained and realistic CDCL model. . .

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

What We Want

More fine-grained and realistic CDCL model. . .

- Capture restarts, clause learning, memory management, et cetera
- 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 polynomial blow-ups)
- How does memory management affect proof search quality?
- Do restarts increase reasoning power? (Or just a helpful heuristic?)
- How do other heuristics help or hinder proof search?

What We Get

- Much less impressive results than we would have liked...
  (but these are hard problems)

What We Get

- Much less impressive results than we would have liked...
  (but these are hard problems)
- Present proof system modelling CDCL no-one can complain about*
  (except it's messy to analyse)

What We Get

- Much less impressive results than we would have liked. . .
  (but these are hard problems)
- Present proof system modelling CDCL no-one can complain about*
  (except it's messy to analyse)
- Already known: no clause learning ⇒ collapse to tree-like resolution

What We Get

- Much less impressive results than we would have liked...
  (but these are hard problems)
- Present proof system modelling CDCL no-one can complain about*
  (except it's messy to analyse)
- Already known: no clause learning $\Rightarrow$ collapse to tree-like resolution
- We show too aggressive clause removal can cause exponential blow-up in running time, matching theory [BN11, BBI12, BNT13]

What We Get

- Much less impressive results than we would have liked...
  (but these are hard problems)
- Present proof system modelling CDCL no-one can complain about*
  (except it's messy to analyse)
- Already known: no clause learning $\Rightarrow$ collapse to tree-like resolution
- We show too aggressive clause removal can cause exponential blow-up in running time, matching theory [BN11, BBI12, BNT13]
- Involves time- and space-efficient simulations of some resolution proofs (but far from general simulation result)

What We Get

- Much less impressive results than we would have liked...
  (but these are hard problems)
- Present proof system modelling CDCL no-one can complain about*
  (except it's messy to analyse)
- Already known: no clause learning $\Rightarrow$ collapse to tree-like resolution
- We show too aggressive clause removal can cause exponential blow-up in running time, matching theory [BN11, BBI12, BNT13]
- Involves time- and space-efficient simulations of some resolution proofs (but far from general simulation result)
- These simulations do not need restarts (impossible to prove in principle for model in [AFT11, PD11])

What We Get

- Much less impressive results than we would have liked...
  (but these are hard problems)
- Present proof system modelling CDCL no-one can complain about*
  (except it's messy to analyse)
- Already known: no clause learning ⇒ collapse to tree-like resolution
- We show too aggressive clause removal can cause exponential blow-up in running time, matching theory [BN11, BBI12, BNT13]
- Involves time- and space-efficient simulations of some resolution proofs (but far from general simulation result)
- These simulations do not need restarts (impossible to prove in principle for model in [AFT11, PD11])

(*) So if you see any issues with the model, we definitely want to know
Obviously, must abstract away some features, but we feel we capture the essentials

Some Notation and Terminology

- Literal a: variable x or its negation $\overline{x}$ (or $\neg x$)
- Clause $C = a_1 \vee \cdots \vee a_k$: disjunction of literals
  (Consider as sets, so no repetitions and order irrelevant)
- CNF formula $F = C_1 \wedge \cdots \wedge C_m$: conjunction of clauses
- N denotes size of formula (# literals counted with repetitions)
- $\mathcal{O}(f(N))$ grows at most as quickly as $f(N)$ asymptotically
  $\Omega(g(N))$ grows at least as quickly as $g(N)$ asymptotically

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Proof ends when empty clause $\bot$ derived

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Proof ends when empty clause $\perp$ derived

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

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

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

Proof ends when empty clause $\bot$ derived

Can represent proof/refutation as

- annotated list or
- directed acyclic graph

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

The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

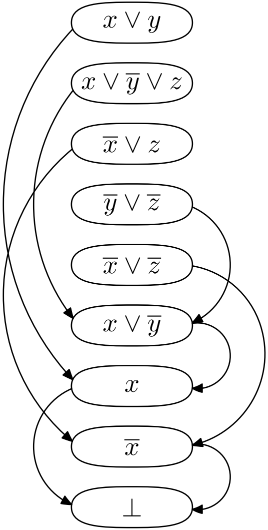
Derive new clauses by resolution rule

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

Proof ends when empty clause $\bot$ derived

Can represent proof/refutation as

- annotated list or
- directed acyclic graph



The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

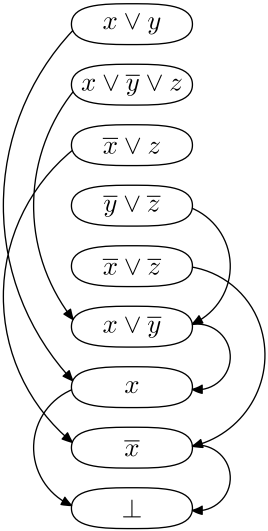
$$\frac{C \lor x \qquad D \lor \overline{x}}{C \lor D}$$

Proof ends when empty clause $\perp$ derived

Can represent proof/refutation as

- annotated list or
- directed acyclic graph

Tree-like if DAG is tree



The Resolution Proof System

Goal: refute **unsatisfiable** CNF

Start with clauses of formula (axioms)

Derive new clauses by resolution rule

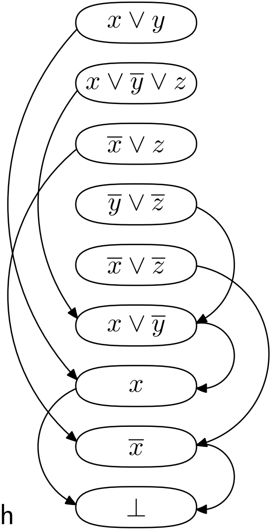
$$\frac{C \lor x \qquad D \lor \overline{x}}{C \lor D}$$

Proof ends when empty clause $\bot$ derived

Can represent proof/refutation as

- annotated list or
- directed acyclic graph

Tree-like if DAG is tree
Regular if resolved variables don't repeat on path



Resolution Size/Length

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

Resolution Size/Length

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

Most fundamental measure in proof complexity

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

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

Matching $\exp(\Omega(N))$ lower bounds known [Urq87, CS88, BW01]

Resolution Space

Space = max # clauses in memory when performing refutation

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

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

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

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

Resolution Space

Space = max # clauses in memory when performing refutation

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

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

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

Example: Space at step 7 ...

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

Resolution Space

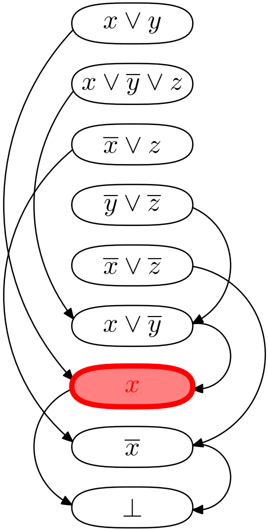
Space = max # clauses in memory when performing refutation

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

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

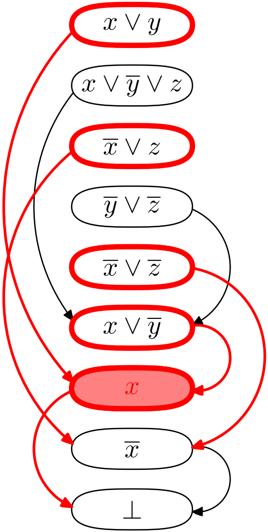
Space = max # clauses in memory when performing refutation

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

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

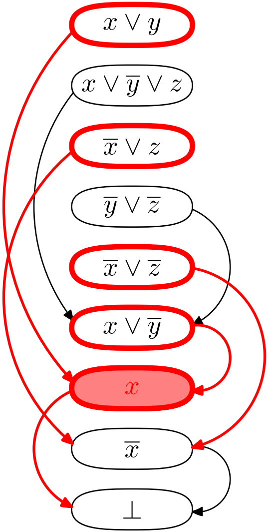
Motivated by SAT solver memory usage (but also intrinsically interesting for proof complexity)

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

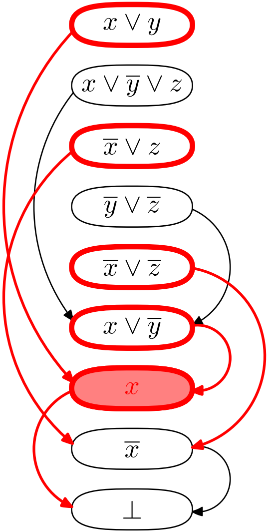
Motivated by SAT solver memory usage (but also intrinsically interesting for proof complexity)

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

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

Example: Space at step 7 is 5

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



Bounds on Resolution Space

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

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

Bounds on Resolution Space

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

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

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

Bounds on Resolution Space

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

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

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

But:

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

Bounds on Resolution Space

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

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

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

But:

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

Which leads to a natural question...

Length-Space Trade-offs

Length $\approx$ running time
Space $\approx$ memory consumption
SAT solvers aggressively try to minimize both — is this possible?

Length-Space Trade-offs

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

Theorem ([BN11, BBI12, BNT13])

There are formulas for which

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

So no meaningful simultaneous optimization possible in worst case

Length-Space Trade-offs

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

Theorem ([BN11, BBI12, BNT13])

There are formulas for which

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

So no meaningful simultaneous optimization possible in worst case
At least for resolution proofs — but what about CDCL proof search?

CDCL Model (1/2)

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

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

CDCL Model (1/2)

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

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

Clause database $\mathcal{D}$: set of initial + learned clauses

CDCL Model (1/2)

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

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

Clause database $\mathcal{D}$: set of initial + learned clauses

Solver starts in **Default** mode and transits to **Conflict**, **Unit**, or **Decision**

CDCL Model (1/2)

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

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

Clause database $\mathcal{D}$: set of initial + learned clauses

Solver starts in **Default** mode and transits to **Conflict**, **Unit**, or **Decision**

Default **If** all variables assigned, output `SAT`;
else if trail falsifies clause $C \in \mathcal{D}$, move to **Conflict**;
else if some $C \in \mathcal{D}$ unit, move to **Unit**;
else solver is in stable state;

CDCL Model (1/2)

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

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

Clause database $\mathcal{D}$: set of initial + learned clauses

Solver starts in **Default** mode and transits to **Conflict**, **Unit**, or **Decision**

Default **If** all variables assigned, output `SAT`;
else if trail falsifies clause $C \in \mathcal{D}$, move to **Conflict**;
else if some $C \in \mathcal{D}$ unit, move to **Unit**;
else solver is in stable state; do in sequence:

1. decide whether to restart, i.e., set trail to $()$;
2. decide whether to apply database reduction to $\mathcal{D}$;
3. move to **Decision**

CDCL Model (2/2)

Unit Arbitrarily pick clause $C \in \mathcal{D}$ unit w.r.t. trail
Add propagated assignment $x \stackrel{C}{=} b$ to trail
Move to **Default**

CDCL Model (2/2)

Unit Arbitrarily pick clause $C \in \mathcal{D}$ unit w.r.t. trail
Add propagated assignment $x \stackrel{C}{=} b$ 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**

CDCL Model (2/2)

Unit Arbitrarily pick clause $C \in \mathcal{D}$ unit w.r.t. trail
Add propagated assignment $x \stackrel{C}{=} b$ 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 $x \stackrel{\mathsf{d}}{=} b$ to trail
Move to **Default**

CDCL Model (2/2)

Unit Arbitrarily pick clause $C \in \mathcal{D}$ unit w.r.t. trail
Add propagated assignment $x \stackrel{C}{=} b$ 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 $x \stackrel{\mathsf{d}}{=} b$ to trail
Move to **Default**

Model draws heavily on [AFT11, PD11]
Combined with ideas from [BHJ08] to capture memory and restarts

CDCL Cannot Do Better than Resolution

Theorem

CDCL with "standard" learning scheme (e.g., UIP) decides F in time τ and space $s \Rightarrow F$ has resolution proof in length $\leq \tau$ and space $\leq s + \mathcal{O}(1)$

CDCL Cannot Do Better than Resolution

Theorem

CDCL with "standard" learning scheme (e.g., UIP) decides F *in time* τ *and space* $s \Rightarrow F$ *has resolution proof in length* $\leq \tau$ *and space* $\leq s + \mathcal{O}(1)$

Fairly obvious for time/length

CDCL Cannot Do Better than Resolution

Theorem

CDCL with "standard" learning scheme (e.g., UIP) decides F in time τ and space $s \Rightarrow F$ has resolution proof in length $\leq \tau$ and space $\leq s + \mathcal{O}(1)$

Fairly obvious for time/length

A priori not so obvious for space
(but proof not hard once one gets the model right)

CDCL Cannot Do Better than Resolution

Theorem

CDCL with "standard" learning scheme (e.g., UIP) decides F *in time* τ *and space* $s \Rightarrow F$ *has resolution proof in length* $\leq \tau$ *and space* $\leq s + \mathcal{O}(1)$

Fairly obvious for time/length

A priori not so obvious for space
(but proof not hard once one gets the model right)

Means that 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 (slightly informal)

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

- *CDCL with 1UIP learning and no restarts can decide F_N in time $\mathcal{O}(N^k)$ and space $\mathcal{O}(N^k)$*
- *CDCL with 1UIP learning and no restarts can decide F_N in space $\mathcal{O}(\log^2 N)$ and time $N^{\mathcal{O}(\log N)}$*
- *For any CDCL run in time τ and space s using any learning scheme and restart policy it holds that $\tau \gtrsim (N^k/s)^{\Omega(\log\log N/\log\log\log N)}$*

Time-Space Trade-Offs for CDCL (in English)

Rephrasing theorem on previous slide to convey high-level message:

- The formulas F_N are somewhat tricky (require more than linear 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)

Rephrasing theorem on previous slide to convey high-level message:

- The formulas F_N are somewhat tricky (require more than linear 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 behaviour for theoretical benchmarks
- But have some indications of similar behaviour for scaled-down versions in practical experiments [ENSS16]

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 $t = 1, 2, 3, \ldots$
- Make sure clause database size $\approx$ space of proof at all times

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 $t = 1, 2, 3, \ldots$
- Make sure clause database size $\approx$ space of proof at all times

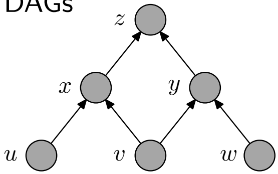
Not as easy as it seems...

- Unit propagation + clause database cause problems
- Suppose have $C \vee x$ and $D \vee \overline{x}$ and now want to learn $C \vee D$
- Easy: decide to make $C \vee D$ false $\Rightarrow$ conflict on x
- But clauses in database can propagate "wrong values"
  $\Rightarrow$ proof search veers off in different direction

Illustrate on One of Benchmarks: Pebbling Formulas

CNF formulas encoding so-called pebble games on DAGs

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

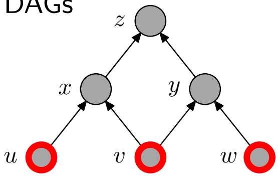


- sources are true
- truth propagates upwards
- but sink is false

Illustrate on One of Benchmarks: Pebbling Formulas

CNF formulas encoding so-called pebble games on DAGs

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. $\neg(z_1 \oplus z_2)$

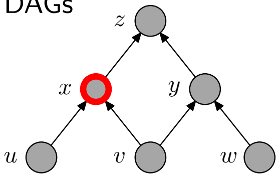


- sources are true
- truth propagates upwards
- but sink is false

Illustrate on One of Benchmarks: Pebbling Formulas

CNF formulas encoding so-called pebble games on DAGs

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

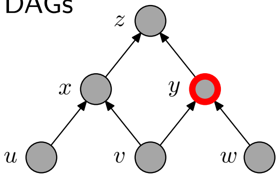


- sources are true
- truth propagates upwards
- but sink is false

Illustrate on One of Benchmarks: Pebbling Formulas

CNF formulas encoding so-called pebble games on DAGs

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

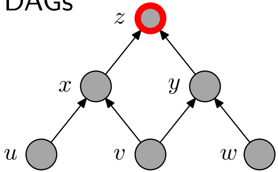


- sources are true
- truth propagates upwards
- but sink is false

Illustrate on One of Benchmarks: Pebbling Formulas

CNF formulas encoding so-called pebble games on DAGs

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

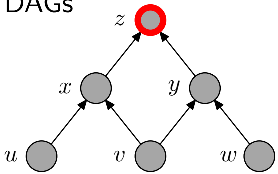


- sources are true
- truth propagates upwards
- but sink is false

Illustrate on One of Benchmarks: Pebbling Formulas

CNF formulas encoding so-called pebble games on DAGs

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. $\neg(z_1 \oplus z_2)$

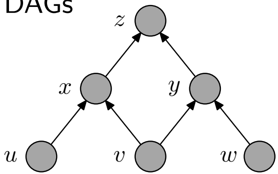


- sources are true
- truth propagates upwards
- but sink is false

Illustrate on One of Benchmarks: Pebbling Formulas

CNF formulas encoding so-called pebble games on DAGs

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



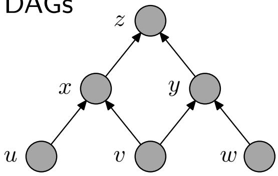
- sources are true
- truth propagates upwards
- but sink is false

Rewrite, e.g., $(x_1 \oplus x_2) \rightarrow (y_1 \oplus y_2)$ in CNF as

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

Illustrate on One of Benchmarks: Pebbling Formulas

CNF formulas encoding so-called pebble games on DAGs



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

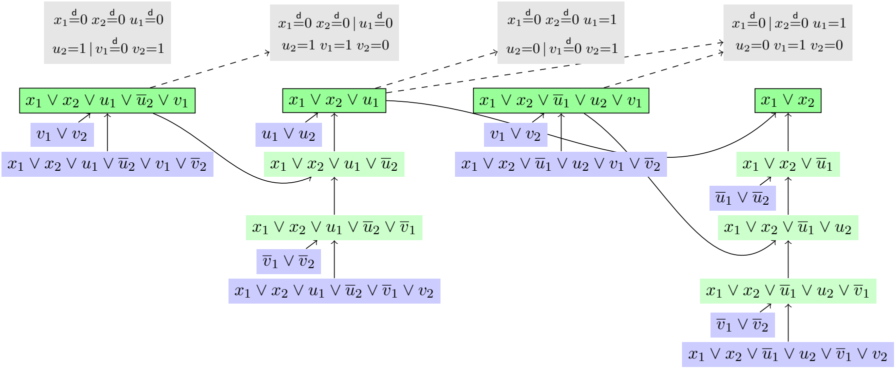
- sources are true
- truth propagates upwards
- but sink is false

Rewrite, e.g., $(x_1 \oplus x_2) \rightarrow (y_1 \oplus y_2)$ in CNF as

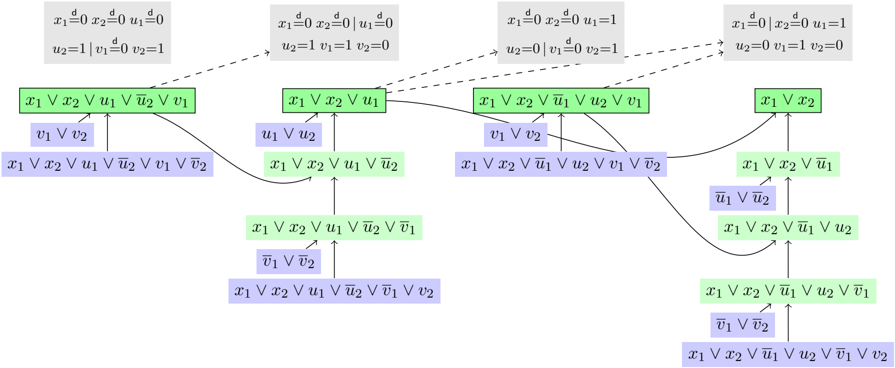
$$(x_1 \vee \overline{x}_2 \vee y_1 \vee y_2) \wedge (x_1 \vee \overline{x}_2 \vee \overline{y}_1 \vee \overline{y}_2)$$
$$\wedge\ (\overline{x}_1 \vee x_2 \vee y_1 \vee y_2) \wedge (\overline{x}_1 \vee x_2 \vee \overline{y}_1 \vee \overline{y}_2)$$

Pebble game trade-offs ⇒ resolution size-space trade-offs [BN08, BN11]

Why Life Without Restarts Might Be Tricky

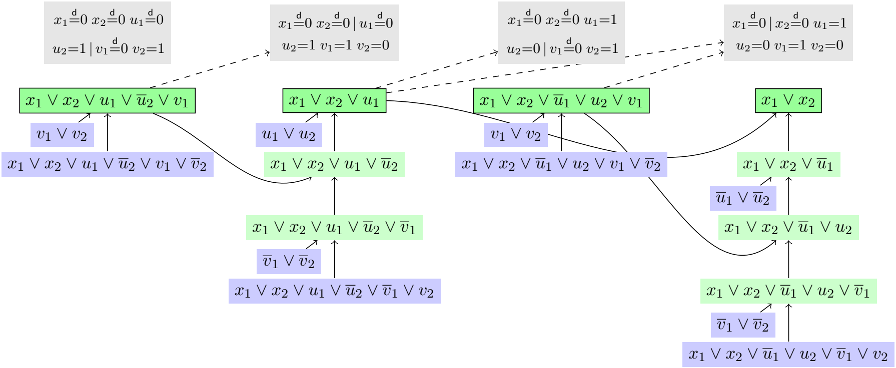


Why Life Without Restarts Might Be Tricky



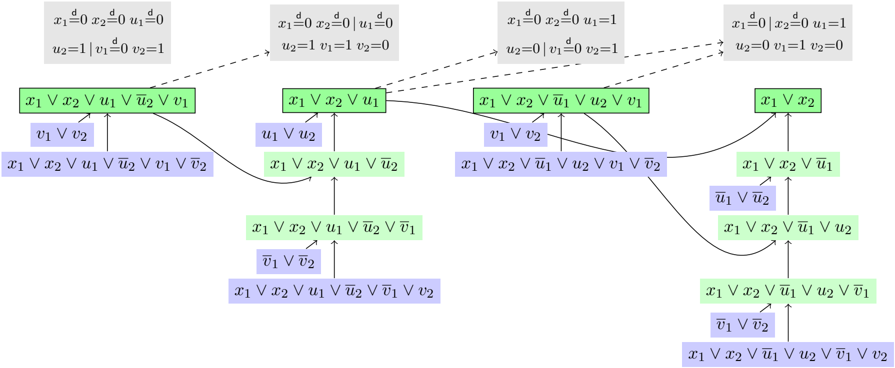
- Know $u_1 \oplus u_2$ and $v_1 \oplus v_2$; want to learn $x_1 \oplus x_2$

Why Life Without Restarts Might Be Tricky



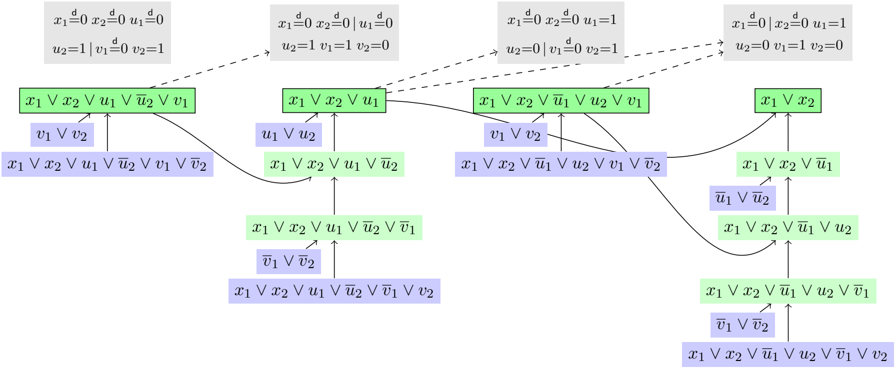
- Know $u_1 \oplus u_2$ and $v_1 \oplus v_2$; want to learn $x_1 \oplus x_2$
- Decide $x_1 = x_2 = 0$ ⇒ easy to learn $x_1 \lor x_2$ — so far, so good

Why Life Without Restarts Might Be Tricky



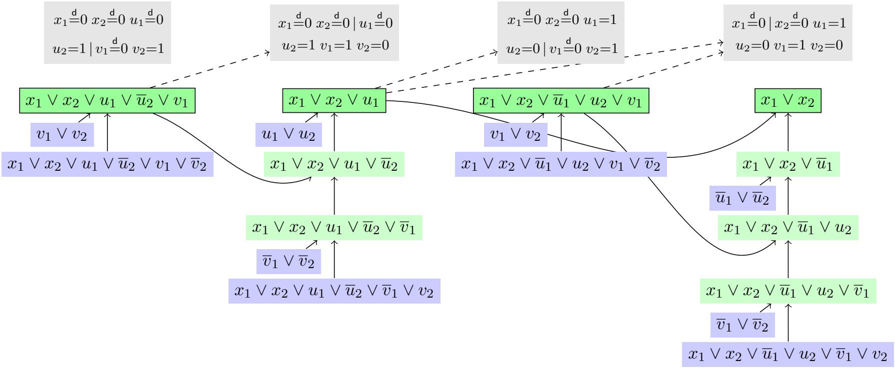
- Know $u_1 \oplus u_2$ and $v_1 \oplus v_2$; want to learn $x_1 \oplus x_2$
- Decide $x_1 = x_2 = 0$ $\Rightarrow$ easy to learn $x_1 \lor x_2$ — so far, so good
- $x_1 \lor x_2$ asserts $x_2 = 1$ but $x_1 = 0$ left on trail $\Rightarrow$ $x_1 \oplus x_2$ true

Why Life Without Restarts Might Be Tricky



- Know $u_1 \oplus u_2$ and $v_1 \oplus v_2$; want to learn $x_1 \oplus x_2$
- Decide $x_1 = x_2 = 0$ $\Rightarrow$ easy to learn $x_1 \lor x_2$ — so far, so good
- $x_1 \lor x_2$ asserts $x_2 = 1$ but $x_1 = 0$ left on trail $\Rightarrow$ $x_1 \oplus x_2$ true
- Need to erase decision $x_1 = 0$ from trail to learn $\overline{x}_1 \lor \overline{x}_2$

Why Life Without Restarts Might Be Tricky



- Know $u_1 \oplus u_2$ and $v_1 \oplus v_2$; want to learn $x_1 \oplus x_2$
- Decide $x_1 = x_2 = 0 \Rightarrow$ easy to learn $x_1 \lor x_2$ — so far, so good
- $x_1 \lor x_2$ asserts $x_2 = 1$ but $x_1 = 0$ left on trail $\Rightarrow x_1 \oplus x_2$ true
- Need to erase decision $x_1 = 0$ from trail to learn $\overline{x}_1 \lor \overline{x}_2$
- Easy with restarts — major pain without…

Open Problems

CDCL vs. resolution

- Can CDCL simulate resolution time- and space-efficiently in theory?
- Is CDCL competitive with resolution in practice?

Open Problems

CDCL vs. resolution

- Can CDCL simulate resolution time- and space-efficiently in theory?
- Is CDCL competitive with resolution in practice?

Importance of restarts

- Is CDCL without restarts strictly weaker than resolution?
- Failed separation attempts in [BHJ08, BBJ14, BK14, BS14] for formulas hard for regular resolution
- But models of CDCL too strong! No real practical implications
- $[\text{CDCL} + (\text{standard heuristics}) - \text{restarts}]$ weaker than resolution?

Open Problems

CDCL vs. resolution

- Can CDCL simulate resolution time- and space-efficiently in theory?
- Is CDCL competitive with resolution in practice?

Importance of restarts

- Is CDCL without restarts strictly weaker than resolution?
- Failed separation attempts in [BHJ08, BBJ14, BK14, BS14] for formulas hard for regular resolution
- But models of CDCL too strong! No real practical implications
- $[\text{CDCL} + (\text{standard heuristics}) - \text{restarts}]$ weaker than resolution?

Theoretical study of power (or weakness) of other heuristics

- How do other heuristics help or hinder proof search?
- Does LBD (literal block distance) measure identify important clauses?
- Prove that VSIDS (variable state independent decaying sum) sometimes goes terribly wrong? (See this on some theory benchmarks)

Summing up This Presentation

This work part of larger effort to connect proof complexity and SAT solving (see survey paper [Nor15] for wider context)

Summing up This Presentation

This work part of larger effort to connect proof complexity and SAT solving (see survey paper [Nor15] for wider context)

Our contributions:

- Fine-grained model of CDCL as proof system
- Time-space trade-offs for CDCL proof search

Summing up This Presentation

This work part of larger effort to connect proof complexity and SAT solving (see survey paper [Nor15] for wider context)

Our contributions:

- Fine-grained model of CDCL as proof system
- Time-space trade-offs for CDCL proof search

Some open problems (not exhaustive list):

- Can CDCL simulate resolution time- and space-efficiently?
- Understand better the role of restarts
- Prove limitations of CDCL with current state-of-the-art heuristics(?)

Summing up This Presentation

This work part of larger effort to connect proof complexity and SAT solving (see survey paper [Nor15] for wider context)

Our contributions:

- Fine-grained model of CDCL as proof system
- Time-space trade-offs for CDCL proof search

Some open problems (not exhaustive list):

- Can CDCL simulate resolution time- and space-efficiently?
- Understand better the role of restarts
- Prove limitations of CDCL with current state-of-the-art heuristics(?)

Thank you for your attention!

References I

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

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

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

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

[BBJ14] Maria Luisa Bonet, Sam Buss, and Jan Johannsen. Improved separations of regular resolution from clause learning proof systems. *Journal of Artificial Intelligence Research*, 49:669–703, 2014.

References II

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

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

[BK14] Samuel R. Buss and Leszek Kołodziejczyk. Small stone in pool. *Logical Methods in Computer Science*, 10, June 2014.

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

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

References III

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

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

[BS14] Paul Beame and Ashish Sabharwal. Non-restarting SAT solvers with simple preprocessing can efficiently simulate resolution. In *Proceedings of the 28th National Conference on Artificial Intelligence (AAAI '14)*, pages 2608–2615. AAAI Press, July 2014.

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

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

References IV

[ENSS16] Jan Elffers, Jakob Nordström, Laurent Simon, and Karem A. Sakallah. Seeking practical CDCL insights from theoretical SAT benchmarks. Manuscript in preparation, 2016.

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

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

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

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

References V

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

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

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

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