Relating Proof Complexity Measures and Practical Hardness of SAT

Jakob Nordström

KTH Royal Institute of Technology
Stockholm, Sweden

First Symposium on Structure in Hard Combinatorial Problems
Vienna, Austria
May 16–18, 2013

Joint work with Matti Järvisalo, Massimo Lauria, Arie Matsliah, Marc Vinyals, and Stanislav Živný

Proof Complexity and SAT Solving

Proof complexity

- Satsifiability fundamental problem in theoretical computer science
- SAT proven NP-complete by Stephen Cook in 1971
- Hence totally intractable in worst case (probably)
- One of the million dollar "Millennium Problems"

Proof Complexity and SAT Solving

Proof complexity

- Satsifiability fundamental problem in theoretical computer science
- SAT proven NP-complete by Stephen Cook in 1971
- Hence totally intractable in worst case (probably)
- One of the million dollar "Millennium Problems"

SAT solving

- Enormous progress in performance last 10–15 years
- State-of-the-art solvers can deal with real-world instances with millions of variables
- But best solvers still based on methods from early 1960s
- Tiny formulas known that are totally beyond reach

Proof Complexity and SAT Solving

Proof complexity

- Satsifiability fundamental problem in theoretical computer science
- SAT proven NP-complete by Stephen Cook in 1971
- Hence totally intractable in worst case (probably)
- One of the million dollar "Millennium Problems"

SAT solving

- Enormous progress in performance last 10–15 years
- State-of-the-art solvers can deal with real-world instances with millions of variables
- But best solvers still based on methods from early 1960s
- Tiny formulas known that are totally beyond reach

What makes formulas hard or easy in practice for SAT solvers?

Proof Complexity and SAT Solving

Proof complexity

- Satsifiability fundamental problem in theoretical computer science
- SAT proven NP-complete by Stephen Cook in 1971
- Hence totally intractable in worst case (probably)
- One of the million dollar "Millennium Problems"

SAT solving

- Enormous progress in performance last 10–15 years
- State-of-the-art solvers can deal with real-world instances with millions of variables
- But best solvers still based on methods from early 1960s
- Tiny formulas known that are totally beyond reach

What makes formulas hard or easy in practice for SAT solvers?

What (if anything) can proof complexity say about this?

Outline

1. SAT solving and Proof Complexity
   - SAT solving and DPLL
   - Proof Complexity and Resolution
   - Our Results
2. Experiments
   - Benchmark Formulas
   - Set-up
   - Results
3. Directions for Future Research

From Proving Tautologies to Disproving CNF Formulas

Conjunctive normal form (CNF)

ANDs of ORs of variables or negated variables
(or conjunctions of disjunctive clauses)

Example:

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

Proving that a formula in propositional logic is **always** satisfied

$\Updownarrow$

Proving that a CNF formula is **never** satisfied

Some Terminology

- Literal a: variable x or its negation $\overline{x}$
- Clause $C = a_1 \vee \cdots \vee a_k$: disjunction of literals
  (Consider as sets, so no repetitions and order irrelevant)
- CNF formula $F = C_1 \wedge \cdots \wedge C_m$: conjunction of clauses
- k-CNF formula: CNF formula with clauses of size $\leq k$
  (where k is some constant)
- All formulas assumed to be k-CNFs in this talk
  (for simplicity of exposition)

The DPLL Method

Based on [Davis & Putnam '60] and [Davis, Logemann & Loveland '62]

Somewhat simplified description:

The DPLL Method

Based on [Davis & Putnam '60] and [Davis, Logemann & Loveland '62]

Somewhat simplified description:

- If F contains an empty clause (without literals), then report "unsatisfiable"

The DPLL Method

Based on [Davis & Putnam '60] and [Davis, Logemann & Loveland '62]

Somewhat simplified description:

- If F contains an empty clause (without literals), then report "unsatisfiable"
- Otherwise pick some variable x in F

The DPLL Method

Based on [Davis & Putnam '60] and [Davis, Logemann & Loveland '62]

Somewhat simplified description:

- If F contains an empty clause (without literals), then report "unsatisfiable"
- Otherwise pick some variable x in F
- Set $x = 0$, simplify F and try to refute recursively

The DPLL Method

Based on [Davis & Putnam '60] and [Davis, Logemann & Loveland '62]

Somewhat simplified description:

- If F contains an empty clause (without literals), then report "unsatisfiable"
- Otherwise pick some variable x in F
- Set $x = 0$, simplify F and try to refute recursively
- Set $x = 1$, simplify F and try to refute recursively

The DPLL Method

Based on [Davis & Putnam '60] and [Davis, Logemann & Loveland '62]

Somewhat simplified description:

- If F contains an empty clause (without literals), then report "unsatisfiable"
- Otherwise pick some variable x in F
- Set $x = 0$, simplify F and try to refute recursively
- Set $x = 1$, simplify F and try to refute recursively
- If result in both cases "unsatisfiable", then report "unsatisfiable"

A DPLL Toy Example

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

A DPLL Toy Example

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

Visualize execution of DPLL algorithm as search tree

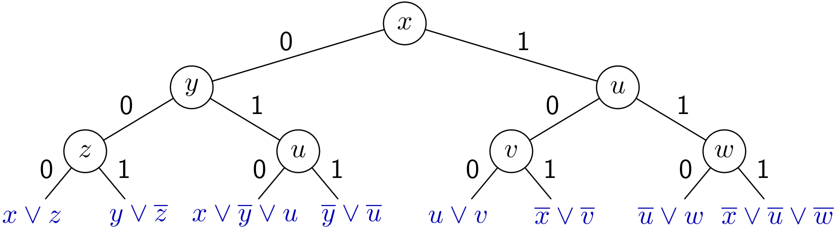
Pick variables in internal nodes; terminate in leaves when falsfied clause found

A DPLL Toy Example

$$
\begin{aligned}
F = \quad & (x \vee z) \wedge (y \vee \overline{z}) \wedge (x \vee \overline{y} \vee u) \wedge (\overline{y} \vee \overline{u}) \\
& \wedge (u \vee v) \wedge (\overline{x} \vee \overline{v}) \wedge (\overline{u} \vee w) \wedge (\overline{x} \vee \overline{u} \vee \overline{w})
\end{aligned}
$$

Visualize execution of DPLL algorithm as search tree

Pick variables in internal nodes; terminate in leaves when falsfied clause found

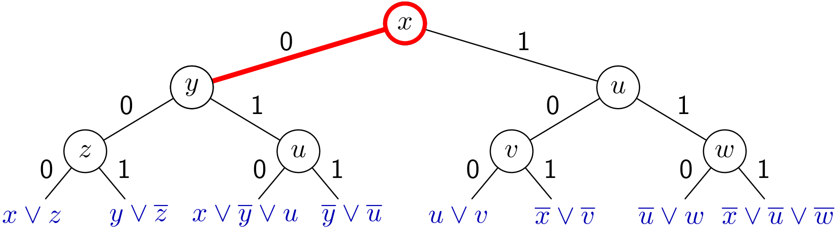


A DPLL Toy Example

$$F = \quad (\qquad z) \land (y \lor \overline{z}) \land (\qquad \overline{y} \lor u) \land (\overline{y} \lor \overline{u})$$
$$\land \; (u \lor v) \land (\overline{x} \lor \overline{v}) \land (\overline{u} \lor w) \land (\overline{x} \lor \overline{u} \lor \overline{w})$$

Visualize execution of DPLL algorithm as search tree

Pick variables in internal nodes; terminate in leaves when falsfied clause found

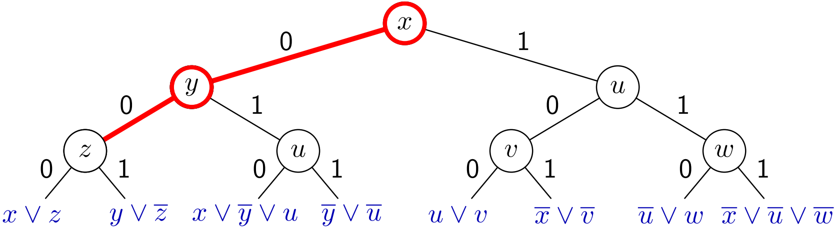


A DPLL Toy Example

$$F = \quad (\quad z) \wedge (\quad \overline{z}) \wedge (\quad \overline{y} \vee u) \wedge (\overline{y} \vee \overline{u})$$
$$\wedge\, (u \vee v) \wedge (\overline{x} \vee \overline{v}) \wedge (\overline{u} \vee w) \wedge (\overline{x} \vee \overline{u} \vee \overline{w})$$

Visualize execution of DPLL algorithm as search tree

Pick variables in internal nodes; terminate in leaves when falsfied clause found

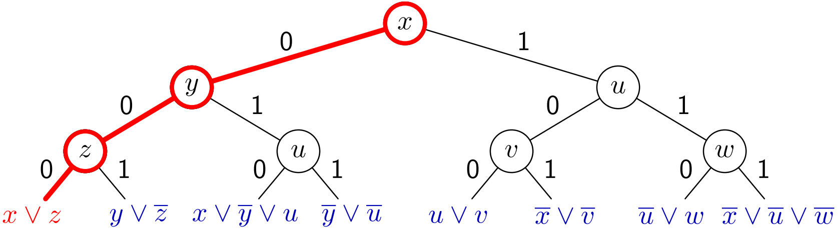


A DPLL Toy Example

$$F = (\quad) \land (\quad \overline{z}) \land (\quad \overline{y} \lor u) \land (\overline{y} \lor \overline{u})$$
$$\land\, (u \lor v) \land (\overline{x} \lor \overline{v}) \land (\overline{u} \lor w) \land (\overline{x} \lor \overline{u} \lor \overline{w})$$

Visualize execution of DPLL algorithm as search tree

Pick variables in internal nodes; terminate in leaves when falsfied clause found

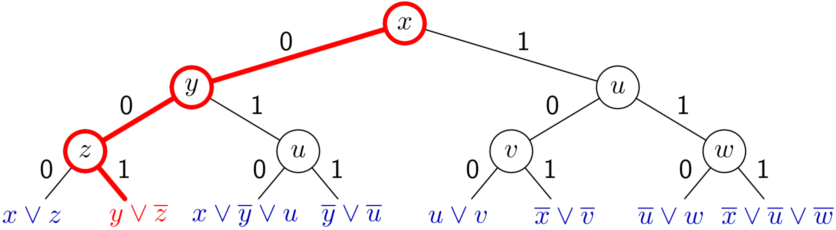


A DPLL Toy Example

$$F = (\quad z) \land (\quad) \land (\quad \overline{y} \lor u) \land (\overline{y} \lor \overline{u}) \land (u \lor v) \land (\overline{x} \lor \overline{v}) \land (\overline{u} \lor w) \land (\overline{x} \lor \overline{u} \lor \overline{w})$$

Visualize execution of DPLL algorithm as search tree

Pick variables in internal nodes; terminate in leaves when falsfied clause found

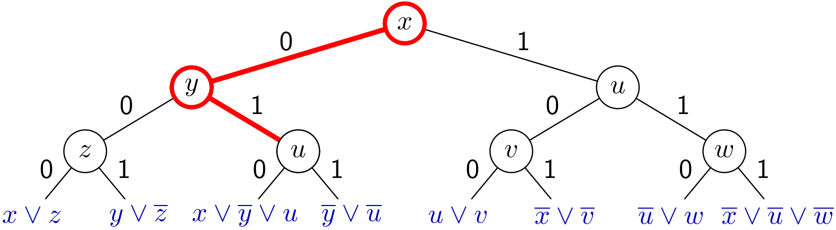


A DPLL Toy Example

$$F = \quad (\quad z) \land (y \lor \overline{z}) \land (\quad u) \land (\quad \overline{u})$$
$$\land\ (u \lor v) \land (\overline{x} \lor \overline{v}) \land (\overline{u} \lor w) \land (\overline{x} \lor \overline{u} \lor \overline{w})$$

Visualize execution of DPLL algorithm as search tree

Pick variables in internal nodes; terminate in leaves when falsfied clause found

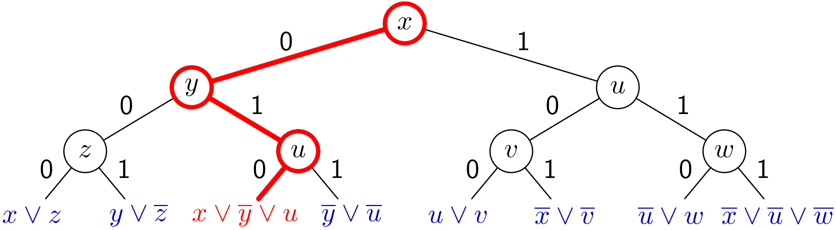


A DPLL Toy Example

$$F = (\quad z) \land (y \lor \overline{z}) \land (\qquad) \land (\quad \overline{u}) \land (\quad v) \land (\overline{x} \lor \overline{v}) \land (\overline{u} \lor w) \land (\overline{x} \lor \overline{u} \lor \overline{w})$$

Visualize execution of DPLL algorithm as search tree

Pick variables in internal nodes; terminate in leaves when falsfied clause found

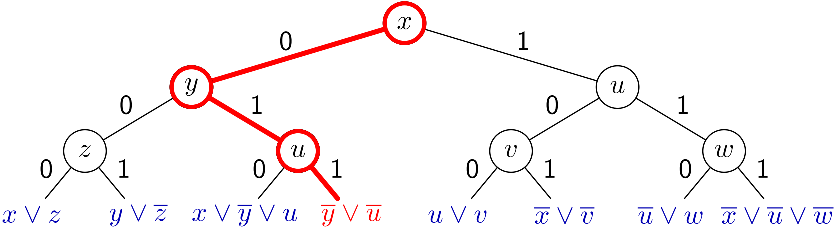


A DPLL Toy Example

$$F = (\quad z) \land (y \lor \overline{z}) \land (\quad u) \land (\quad) \land (u \lor v) \land (\overline{x} \lor \overline{v}) \land (\quad w) \land (\overline{x} \lor \quad \overline{w})$$

Visualize execution of DPLL algorithm as search tree

Pick variables in internal nodes; terminate in leaves when falsfied clause found

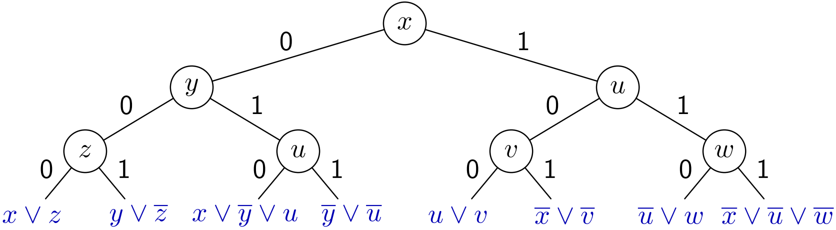


A DPLL Toy Example

$$F = \quad (x \lor z) \land (y \lor \overline{z}) \land (x \lor \overline{y} \lor u) \land (\overline{y} \lor \overline{u}) \\ \land\ (u \lor v) \land (\overline{x} \lor \overline{v}) \land (\overline{u} \lor w) \land (\overline{x} \lor \overline{u} \lor \overline{w})$$

Visualize execution of DPLL algorithm as search tree

Pick variables in internal nodes; terminate in leaves when falsfied clause found



State-of-the-art DPLL SAT solvers

Many more ingredients in modern SAT solvers, for instance:

- Choice of pivot variables crucial
- In particular, always do unit propagation on sole remaining variable in a clause [which our toy example didn't]
- When reaching falsified clause, compute why partial assignment failed — add this info to formula as new clause
  Conflict-driven clause learning (CDCL)
- Every once in a while, restart (but save computed info)

Proof Complexity

Proof search algorithm: defines proof system with derivation rules

Proof complexity: study of proofs in such systems

- Lower bounds: no algorithm can do better (even optimal one always guessing the right move)
- Upper bounds: gives hope for good algorithms if we can search for proofs in system efficiently

Resolution

Resolution rule:

$$\frac{B \vee x \quad C \vee \overline{x}}{B \vee C}$$

Resolution

Resolution rule:

$$\frac{B \vee x \quad C \vee \overline{x}}{B \vee C}$$

Observation

If F is a satisfiable CNF formula and D is derived from clauses $C_1, C_2 \in F$ by the resolution rule, then $F \wedge D$ is satisfiable.

Resolution

Resolution rule:

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

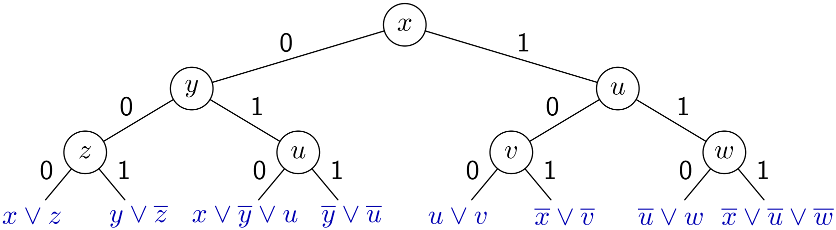
Observation

If F is a satisfiable CNF formula and D is derived from clauses $C_1, C_2 \in F$ by the resolution rule, then $F \land D$ is satisfiable.

Prove F unsatisfiable by deriving the unsatisfiable empty clause $\bot$ from F by resolution

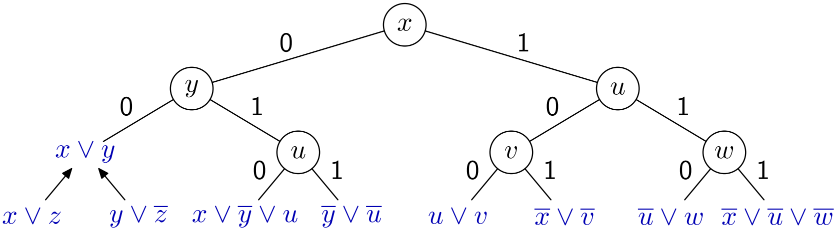
CDCL Solvers Generate Resolution Proofs

Simple example for DPLL:



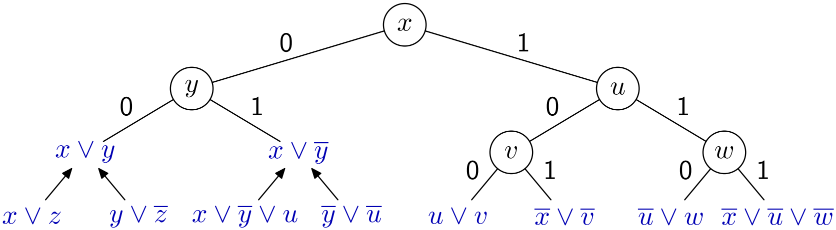
CDCL Solvers Generate Resolution Proofs

Simple example for DPLL:



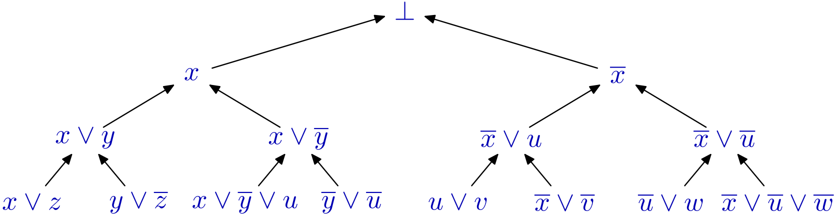
CDCL Solvers Generate Resolution Proofs

Simple example for DPLL:



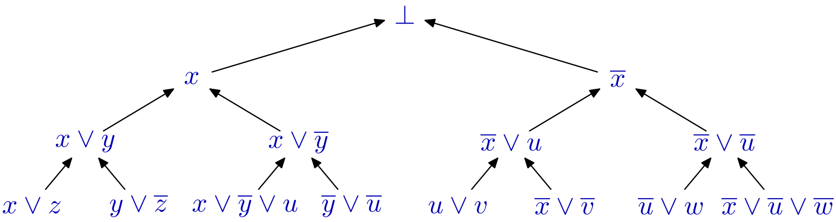
CDCL Solvers Generate Resolution Proofs

Simple example for DPLL:



CDCL Solvers Generate Resolution Proofs

Simple example for DPLL:



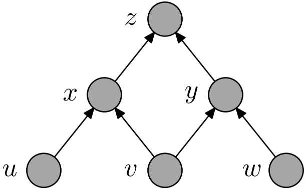
- Conflict-driven clause learning adds “shortcut edges” in tree
- But still yields resolution proof
- True also for (most) preprocessing techniques

The Theoretical Model

- Goal: Refute given CNF formula (i.e., prove it is unsatisfiable)
- Proof system operates with disjunctive clauses
- Proof/refutation is "presented on blackboard"
- Derivation steps:
  - Write down clauses of CNF formula being refuted (axiom clauses)
  - Infer new clauses by resolution rule
  - Erase clauses that are not currently needed (to save space on blackboard)
- Refutation ends when empty clause $\perp$ is derived

Example CNF Formula

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

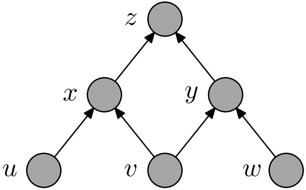


Defined in terms of directed acyclic graph (DAG):

- source vertices true
- truth propagates upwards
- but sink vertex is false

Example CNF Formula

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

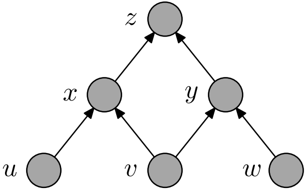


Defined in terms of directed acyclic graph (DAG):

- source vertices true
- truth propagates upwards
- but sink vertex is false

Example CNF Formula

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

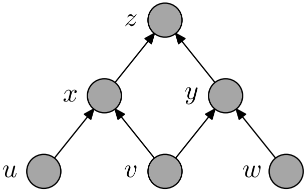


Defined in terms of directed acyclic graph (DAG):

- source vertices true
- truth propagates upwards
- but sink vertex is false

Example CNF Formula

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$



Defined in terms of directed acyclic graph (DAG):

- source vertices true
- truth propagates upwards
- but sink vertex is false

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	0
largest clause seen on board	0
max # lines on board	0

Can write down axioms,
erase used clauses or
infer new clauses by resolution rule
(but only from clauses currently on the board!)

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	1
largest clause seen on board	1
max # lines on board	1

u

Write down axiom 1: u

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	2
largest clause seen on board	1
max # lines on board	2

u
v

Write down axiom 1: u
Write down axiom 2: v

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	3
largest clause seen on board	3
max # lines on board	3

u
v
$\overline{u} \vee \overline{v} \vee x$

Write down axiom 1: u
Write down axiom 2: v
Write down axiom 4: $\overline{u} \vee \overline{v} \vee x$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	3
largest clause seen on board	3
max # lines on board	3

u
v
$\overline{u} \lor \overline{v} \lor x$

Write down axiom 1: u
Write down axiom 2: v
Write down axiom 4: $\overline{u} \lor \overline{v} \lor x$
Infer $\overline{v} \lor x$ from
u and $\overline{u} \lor \overline{v} \lor x$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	4
largest clause seen on board	3
max # lines on board	4

u

v

$\overline{u} \lor \overline{v} \lor x$

$\overline{v} \lor x$

Write down axiom 1: u
Write down axiom 2: v
Write down axiom 4: $\overline{u} \lor \overline{v} \lor x$
Infer $\overline{v} \lor x$ from
u and $\overline{u} \lor \overline{v} \lor x$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	4
largest clause seen on board	3
max # lines on board	4

u
v
$\overline{u} \vee \overline{v} \vee x$
$\overline{v} \vee x$

Write down axiom 2: v
Write down axiom 4: $\overline{u} \vee \overline{v} \vee x$
Infer $\overline{v} \vee x$ from
u and $\overline{u} \vee \overline{v} \vee x$
Erase the clause $\overline{u} \vee \overline{v} \vee x$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	4
largest clause seen on board	3
max # lines on board	4

u

v

$\overline{v} \lor x$

Write down axiom 2: v
Write down axiom 4: $\overline{u} \lor \overline{v} \lor x$
Infer $\overline{v} \lor x$ from
u and $\overline{u} \lor \overline{v} \lor x$
Erase the clause $\overline{u} \lor \overline{v} \lor x$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	4
largest clause seen on board	3
max # lines on board	4

u
v
$\overline{v} \lor x$

Write down axiom 4: $\overline{u} \lor \overline{v} \lor x$
Infer $\overline{v} \lor x$ from
u and $\overline{u} \lor \overline{v} \lor x$
Erase the clause $\overline{u} \lor \overline{v} \lor x$
Erase the clause u

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	4
largest clause seen on board	3
max # lines on board	4

v
$\overline{v} \lor x$

Write down axiom 4: $\overline{u} \lor \overline{v} \lor x$
Infer $\overline{v} \lor x$ from
u and $\overline{u} \lor \overline{v} \lor x$
Erase the clause $\overline{u} \lor \overline{v} \lor x$
Erase the clause u

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	4
largest clause seen on board	3
max # lines on board	4

v
$\overline{v} \lor x$

u and $\overline{u} \lor \overline{v} \lor x$
Erase the clause $\overline{u} \lor \overline{v} \lor x$
Erase the clause u
Infer x from
v and $\overline{v} \lor x$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	5
largest clause seen on board	3
max # lines on board	4

v
$\overline{v} \lor x$
x

u and $\overline{u} \lor \overline{v} \lor x$
Erase the clause $\overline{u} \lor \overline{v} \lor x$
Erase the clause u
Infer x from
v and $\overline{v} \lor x$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	5
largest clause seen on board	3
max # lines on board	4

Board:

v

$\overline{v} \vee x$

x

Erase the clause $\overline{u} \vee \overline{v} \vee x$

Erase the clause u

Infer x from

v and $\overline{v} \vee x$

Erase the clause $\overline{v} \vee x$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	5
largest clause seen on board	3
max # lines on board	4

v
x

Erase the clause $\overline{u} \vee \overline{v} \vee x$
Erase the clause u
Infer x from
  v and $\overline{v} \vee x$
Erase the clause $\overline{v} \vee x$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	5
largest clause seen on board	3
max # lines on board	4

v

x

Erase the clause u
Infer x from
v and $\overline{v} \lor x$
Erase the clause $\overline{v} \lor x$
Erase the clause v

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	5
largest clause seen on board	3
max # lines on board	4

Board:

x

Erase the clause u

Infer x from

v and $\overline{v} \lor x$

Erase the clause $\overline{v} \lor x$

Erase the clause v

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	6
largest clause seen on board	3
max # lines on board	4

Board:

x

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

Infer x from

v and $\overline{v} \lor x$

Erase the clause $\overline{v} \lor x$

Erase the clause v

Write down axiom 6: $\overline{x} \lor \overline{y} \lor z$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	6
largest clause seen on board	3
max # lines on board	4

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

Erase the clause $\overline{v} \vee x$
Erase the clause v
Write down axiom 6: $\overline{x} \vee \overline{y} \vee z$
Infer $\overline{y} \vee z$ from
x and $\overline{x} \vee \overline{y} \vee z$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	7
largest clause seen on board	3
max # lines on board	4

x

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

$\overline{y} \lor z$

Erase the clause $\overline{v} \lor x$

Erase the clause v

Write down axiom 6: $\overline{x} \lor \overline{y} \lor z$

Infer $\overline{y} \lor z$ from

x and $\overline{x} \lor \overline{y} \lor z$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	7
largest clause seen on board	3
max # lines on board	4

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

Erase the clause v
Write down axiom 6: $\overline{x} \lor \overline{y} \lor z$
Infer $\overline{y} \lor z$ from
x and $\overline{x} \lor \overline{y} \lor z$
Erase the clause $\overline{x} \lor \overline{y} \lor z$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	7
largest clause seen on board	3
max # lines on board	4

Board:

x

$\overline{y} \lor z$

Erase the clause v

Write down axiom 6: $\overline{x} \lor \overline{y} \lor z$

Infer $\overline{y} \lor z$ from

x and $\overline{x} \lor \overline{y} \lor z$

Erase the clause $\overline{x} \lor \overline{y} \lor z$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	7
largest clause seen on board	3
max # lines on board	4

x

$\overline{y} \lor z$

Write down axiom 6: $\overline{x} \lor \overline{y} \lor z$

Infer $\overline{y} \lor z$ from

x and $\overline{x} \lor \overline{y} \lor z$

Erase the clause $\overline{x} \lor \overline{y} \lor z$

Erase the clause x

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	7
largest clause seen on board	3
max # lines on board	4

$\overline{y} \lor z$

Write down axiom 6: $\overline{x} \lor \overline{y} \lor z$

Infer $\overline{y} \lor z$ from

x and $\overline{x} \lor \overline{y} \lor z$

Erase the clause $\overline{x} \lor \overline{y} \lor z$

Erase the clause x

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	8
largest clause seen on board	3
max # lines on board	4

$\overline{y} \vee z$
$\overline{v} \vee \overline{w} \vee y$

Infer $\overline{y} \vee z$ from
x and $\overline{x} \vee \overline{y} \vee z$
Erase the clause $\overline{x} \vee \overline{y} \vee z$
Erase the clause x
Write down axiom 5: $\overline{v} \vee \overline{w} \vee y$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	8
largest clause seen on board	3
max # lines on board	4

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

Erase the clause $\overline{x} \lor \overline{y} \lor z$
Erase the clause x
Write down axiom 5: $\overline{v} \lor \overline{w} \lor y$
Infer $\overline{v} \lor \overline{w} \lor z$ from
$\overline{y} \lor z$ and $\overline{v} \lor \overline{w} \lor y$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	9
largest clause seen on board	3
max # lines on board	4

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

Erase the clause $\overline{x} \lor \overline{y} \lor z$
Erase the clause x
Write down axiom 5: $\overline{v} \lor \overline{w} \lor y$
Infer $\overline{v} \lor \overline{w} \lor z$ from
$\overline{y} \lor z$ and $\overline{v} \lor \overline{w} \lor y$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	9
largest clause seen on board	3
max # lines on board	4

$\overline{y} \lor z$

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

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

Erase the clause x

Write down axiom 5: $\overline{v} \lor \overline{w} \lor y$

Infer $\overline{v} \lor \overline{w} \lor z$ from

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

Erase the clause $\overline{v} \lor \overline{w} \lor y$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	9
largest clause seen on board	3
max # lines on board	4

$\overline{y} \lor z$

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

Erase the clause x

Write down axiom 5: $\overline{v} \lor \overline{w} \lor y$

Infer $\overline{v} \lor \overline{w} \lor z$ from

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

Erase the clause $\overline{v} \lor \overline{w} \lor y$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	9
largest clause seen on board	3
max # lines on board	4

$\overline{y} \lor z$

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

Write down axiom 5: $\overline{v} \lor \overline{w} \lor y$

Infer $\overline{v} \lor \overline{w} \lor z$ from

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

Erase the clause $\overline{v} \lor \overline{w} \lor y$

Erase the clause $\overline{y} \lor z$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	9
largest clause seen on board	3
max # lines on board	4

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

Write down axiom 5: $\overline{v} \vee \overline{w} \vee y$
Infer $\overline{v} \vee \overline{w} \vee z$ from
$\overline{y} \vee z$ and $\overline{v} \vee \overline{w} \vee y$
Erase the clause $\overline{v} \vee \overline{w} \vee y$
Erase the clause $\overline{y} \vee z$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	10
largest clause seen on board	3
max # lines on board	4

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

v

Infer $\overline{v} \lor \overline{w} \lor z$ from
$\overline{y} \lor z$ and $\overline{v} \lor \overline{w} \lor y$
Erase the clause $\overline{v} \lor \overline{w} \lor y$
Erase the clause $\overline{y} \lor z$
Write down axiom 2: v

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	11
largest clause seen on board	3
max # lines on board	4

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

$\overline{y} \vee z$ and $\overline{v} \vee \overline{w} \vee y$
Erase the clause $\overline{v} \vee \overline{w} \vee y$
Erase the clause $\overline{y} \vee z$
Write down axiom 2: v
Write down axiom 3: w

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	12
largest clause seen on board	3
max # lines on board	4

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

Erase the clause $\overline{v} \lor \overline{w} \lor y$
Erase the clause $\overline{y} \lor z$
Write down axiom 2: v
Write down axiom 3: w
Write down axiom 7: $\overline{z}$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	12
largest clause seen on board	3
max # lines on board	4

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

Write down axiom 2: v
Write down axiom 3: w
Write down axiom 7: $\overline{z}$
Infer $\overline{w} \lor z$ from
v and $\overline{v} \lor \overline{w} \lor z$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	13
largest clause seen on board	3
max # lines on board	5

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

Write down axiom 2: v
Write down axiom 3: w
Write down axiom 7: $\overline{z}$
Infer $\overline{w} \lor z$ from
v and $\overline{v} \lor \overline{w} \lor z$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	13
largest clause seen on board	3
max # lines on board	5

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

Write down axiom 3: w
Write down axiom 7: $\overline{z}$
Infer $\overline{w} \lor z$ from
v and $\overline{v} \lor \overline{w} \lor z$
Erase the clause v

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	13
largest clause seen on board	3
max # lines on board	5

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

Write down axiom 3: w
Write down axiom 7: $\overline{z}$
Infer $\overline{w} \vee z$ from
v and $\overline{v} \vee \overline{w} \vee z$
Erase the clause v

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	13
largest clause seen on board	3
max # lines on board	5

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

Write down axiom 7: $\overline{z}$
Infer $\overline{w} \lor z$ from
v and $\overline{v} \lor \overline{w} \lor z$
Erase the clause v
Erase the clause $\overline{v} \lor \overline{w} \lor z$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	13
largest clause seen on board	3
max # lines on board	5

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

Write down axiom 7: $\overline{z}$
Infer $\overline{w} \vee z$ from
v and $\overline{v} \vee \overline{w} \vee z$
Erase the clause v
Erase the clause $\overline{v} \vee \overline{w} \vee z$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	13
largest clause seen on board	3
max # lines on board	5

w

$\overline{z}$

$\overline{w} \lor z$

v and $\overline{v} \lor \overline{w} \lor z$

Erase the clause v

Erase the clause $\overline{v} \lor \overline{w} \lor z$

Infer z from

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

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	14
largest clause seen on board	3
max # lines on board	5

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

v and $\overline{v} \lor \overline{w} \lor z$
Erase the clause v
Erase the clause $\overline{v} \lor \overline{w} \lor z$
Infer z from
w and $\overline{w} \lor z$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	14
largest clause seen on board	3
max # lines on board	5

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

Erase the clause v
Erase the clause $\overline{v} \lor \overline{w} \lor z$
Infer z from
w and $\overline{w} \lor z$
Erase the clause w

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	14
largest clause seen on board	3
max # lines on board	5

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

Erase the clause v
Erase the clause $\overline{v} \lor \overline{w} \lor z$
Infer z from
w and $\overline{w} \lor z$
Erase the clause w

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	14
largest clause seen on board	3
max # lines on board	5

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

Erase the clause $\overline{v} \lor \overline{w} \lor z$
Infer z from
w and $\overline{w} \lor z$
Erase the clause w
Erase the clause $\overline{w} \lor z$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	14
largest clause seen on board	3
max # lines on board	5

$\overline{z}$

z

Erase the clause $\overline{v} \vee \overline{w} \vee z$
Infer z from
w and $\overline{w} \vee z$
Erase the clause w
Erase the clause $\overline{w} \vee z$

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	14
largest clause seen on board	3
max # lines on board	5

$\overline{z}$

z

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

Erase the clause w

Erase the clause $\overline{w} \vee z$

Infer $\perp$ from

$\overline{z}$ and z

Example Resolution Refutation

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

Blackboard bookkeeping	
total # clauses on board	15
largest clause seen on board	3
max # lines on board	5

$\overline{z}$

z

$\bot$

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

Erase the clause w

Erase the clause $\overline{w} \lor z$

Infer $\bot$ from

$\overline{z}$ and z

Complexity Measures for Resolution

Let n = size of formula

Length

clauses in refutation — at most $\exp(n)$ [in our example: 15]

Width

Size of largest clause in refutation — at most n [in our example: 3]

Space

Max # clauses one needs to remember when "verifying correctness of refutation on blackboard" — at most n (!) [in our example: 5]

Length

- Clearly lower bound on running time for any CDCL algorithm

Length

- Clearly lower bound on running time for any CDCL algorithm
- But if there is a short refutation, not clear how to find it

Length

- Clearly lower bound on running time for any CDCL algorithm
- But if there is a short refutation, not clear how to find it
- In fact, probably intractable [Aleknovich & Razborov '01]

Length

- Clearly lower bound on running time for any CDCL algorithm
- But if there is a short refutation, not clear how to find it
- In fact, probably intractable [Aleknovich & Razborov '01]
- So small length upper bound might be much too optimistic

Length

- Clearly lower bound on running time for any CDCL algorithm
- But if there is a short refutation, not clear how to find it
- In fact, probably intractable [Aleknovich & Razborov '01]
- So small length upper bound might be much too optimistic
- Not the right measure of "hardness in practice"

Length vs. Width

- Searching for small width refutations known heuristic in AI community

Length vs. Width

- Searching for small width refutations known heuristic in AI community
- Small width ⇒ small length (by counting)

Length vs. Width

- Searching for small width refutations known heuristic in AI community
- Small width $\Rightarrow$ small length (by counting)
- But small length does not necessary imply small width — can have $\sqrt{n}$ width and linear length [Bonet & Galesi '99]

Length vs. Width

- Searching for small width refutations known heuristic in AI community
- Small width ⇒ small length (by counting)
- But small length does not necessary imply small width — can have $\sqrt{n}$ width and linear length [Bonet & Galesi '99]
- So width stricter hardness measure than length

Length vs. Width

- Searching for small width refutations known heuristic in AI community
- Small width ⇒ small length (by counting)
- But small length does not necessary imply small width — can have $\sqrt{n}$ width and linear length [Bonet & Galesi '99]
- So width stricter hardness measure than length
- However, really large (e.g., linear) width implies really large (exponential) length [Ben-Sasson & Wigderson '99]

Length vs. Width

- Searching for small width refutations known heuristic in AI community
- Small width ⇒ small length (by counting)
- But small length does not necessary imply small width — can have $\sqrt{n}$ width and linear length [Bonet & Galesi '99]
- So width stricter hardness measure than length
- However, really large (e.g., linear) width implies really large (exponential) length [Ben-Sasson & Wigderson '99]
- Small width ⇒ CDCL solver will provably be fast
  [Atserias, Fichte & Thurley '09]
  (but slighly idealized theoretical model)

Length vs. Width

- Searching for small width refutations known heuristic in AI community
- Small width ⇒ small length (by counting)
- But small length does not necessary imply small width — can have $\sqrt{n}$ width and linear length [Bonet & Galesi '99]
- So width stricter hardness measure than length
- However, really large (e.g., linear) width implies really large (exponential) length [Ben-Sasson & Wigderson '99]
- Small width ⇒ CDCL solver will provably be fast
  [Atserias, Fichte & Thurley '09]
  (but slighly idealized theoretical model)
- Right hardness measure?

Width vs. Space

- In practice, memory consumption is a very important bottleneck for SAT solvers

Width vs. Space

- In practice, memory consumption is a very important bottleneck for SAT solvers
- So maybe space complexity can be relevant hardness measure?

Width vs. Space

- In practice, memory consumption is a very important bottleneck for SAT solvers
- So maybe space complexity can be relevant hardness measure?
- Space $\geq$ width [Atserias & Dalmau '03]

Width vs. Space

- In practice, memory consumption is a very important bottleneck for SAT solvers
- So maybe space complexity can be relevant hardness measure?
- Space $\geq$ width [Atserias & Dalmau '03]
- But small width does not say **anything** about space [N. '06], [N. & Håstad '08], [Ben-Sasson & N. '08]

Width vs. Space

- In practice, memory consumption is a very important bottleneck for SAT solvers
- So maybe space complexity can be relevant hardness measure?
- Space $\geq$ width [Atserias & Dalmau '03]
- But small width does not say **anything** about space [N. '06], [N. & Håstad '08], [Ben-Sasson & N. '08]
- So space stricter hardness measure than width (but space model even more idealized)

Space vs. Tree-like Space

- Tree-like resolution: Only use each clause once
  Have to rederive from scratch if needed again

Space vs. Tree-like Space

- Tree-like resolution: Only use each clause once
  Have to rederive from scratch if needed again
- Tree-like space: Usual space measure but restricted to such proofs

Space vs. Tree-like Space

- Tree-like resolution: Only use each clause once
  Have to rederive from scratch if needed again
- Tree-like space: Usual space measure but restricted to such proofs
- Proposed as practical measure of hardness of SAT instances in [Ansótegui, Bonet, Levy & Manyà '08]

Space vs. Tree-like Space

- Tree-like resolution: Only use each clause once
  Have to rederive from scratch if needed again
- Tree-like space: Usual space measure but restricted to such proofs
- Proposed as practical measure of hardness of SAT instances in [Ansótegui, Bonet, Levy & Manyà '08]
- Clearly tree-like space $\geq$ space but not known to be different

Space vs. Tree-like Space

- Tree-like resolution: Only use each clause once
  Have to rederive from scratch if needed again
- Tree-like space: Usual space measure but restricted to such proofs
- Proposed as practical measure of hardness of SAT instances in [Ansótegui, Bonet, Levy & Manyà '08]
- Clearly tree-like space $\geq$ space but not known to be different

This work can be viewed as implementing program outlined in [ABLM08]

Result 1: Separation of Space and Tree-like Space

We don't believe in tree-like space as hardness measure

- Tree-like space tightly connected with tree-like length
- Corresponds to DPLL without clause learning
- Would suggest CDCL doesn't buy you anything

Result 1: Separation of Space and Tree-like Space

We don't believe in tree-like space as hardness measure

- Tree-like space tightly connected with tree-like length
- Corresponds to DPLL without clause learning
- Would suggest CDCL doesn't buy you anything

We prove first asymptotic separation of space and tree-like space

Theorem

There are formulas requiring space $\mathcal{O}(1)$ for which tree-like space grows like $\Omega(\log n)$

Only constant-factor separation known before [Esteban & Torán '03]

Result 2: Small Backdoor Sets Imply Small Space

- Backdoor sets: practically motivated hardness measure
- First studied in [Williams, Gomes & Selman '03]
- Real-world SAT instances often have small backdoors

Result 2: Small Backdoor Sets Imply Small Space

- Backdoor sets: practically motivated hardness measure
- First studied in [Williams, Gomes & Selman '03]
- Real-world SAT instances often have small backdoors

We show connections between (strong) backdoors and space complexity (elaborating on [ABLM08])

Theorem (Informal)

If a formula has a small backdoor set (for some common flavours of backdoors), then it requires small space

Result 3: Hardness in Practice Correlates with Space

Recall

$$\log \text{length} \leq \text{width} \leq \text{space} \leq \text{tree-like space}$$

Result 3: Hardness in Practice Correlates with Space

Recall

$$\log \text{length} \leq \text{width} \leq \text{space} \leq \text{tree-like space}$$

Width and space seem like most promising hardness candidates

Result 3: Hardness in Practice Correlates with Space

Recall

$$\log \text{length} \leq \text{width} \leq \text{space} \leq \text{tree-like space}$$

Width and space seem like most promising hardness candidates

Run experiments on formulas with fixed complexity w.r.t. width (and length) but varying space*

- Is running time essentially the same?
- Or does it increase with increasing space?

Result 3: Hardness in Practice Correlates with Space

Recall

$$\log \text{length} \leq \text{width} \leq \text{space} \leq \text{tree-like space}$$

Width and space seem like most promising hardness candidates

Run experiments on formulas with fixed complexity w.r.t. width (and length) but varying space*

- Is running time essentially the same?
- Or does it increase with increasing space?

Experimental results

Running times seem to be somewhat correlated with space complexity**

Result 3: Hardness in Practice Correlates with Space

Recall

$$\log \text{length} \leq \text{width} \leq \text{space} \leq \text{tree-like space}$$

Width and space seem like most promising hardness candidates

Run experiments on formulas with fixed complexity w.r.t. width (and length) but varying space*

- Is running time essentially the same?
- Or does it increase with increasing space?

Experimental results

Running times seem to be somewhat correlated with space complexity**

(*) But such formulas are nontrivial to find
(**) With some caveats to be discussed later

How to Get Hold of Good Benchmark Formulas?

Questions about space complexity and time-space trade-offs fundamental in theoretical computer science

How to Get Hold of Good Benchmark Formulas?

Questions about space complexity and time-space trade-offs fundamental in theoretical computer science

In particular, well-studied (and well-understood) for pebble games modelling calculations described by DAGs ([Cook & Sethi '76] and others)

- Time needed for calculation: # pebbling moves
- Space needed for calculation: max # pebbles required

How to Get Hold of Good Benchmark Formulas?

Questions about space complexity and time-space trade-offs fundamental in theoretical computer science

In particular, well-studied (and well-understood) for pebble games modelling calculations described by DAGs ([Cook & Sethi '76] and others)

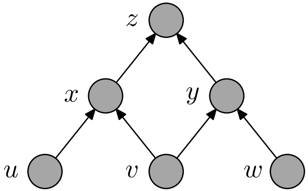
- Time needed for calculation: # pebbling moves
- Space needed for calculation: max # pebbles required

Some quick graph terminology

- DAGs consist of vertices with directed edges between them
- vertices with no incoming edges: sources
- vertices with no outgoing edges: sinks

The Black-White Pebble Game

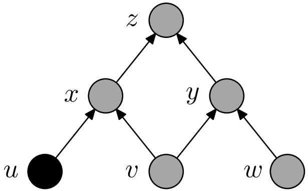
Goal: get single black pebble on sink vertex z of G



# moves	0
Current # pebbles	0
Max # pebbles so far	0

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

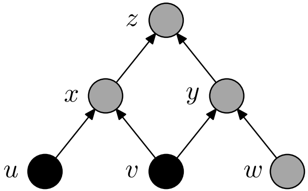


# moves	1
Current # pebbles	1
Max # pebbles so far	1

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

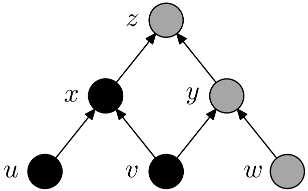


# moves	2
Current # pebbles	2
Max # pebbles so far	2

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

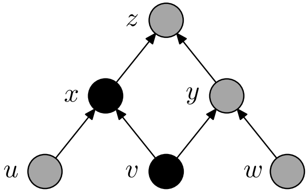


# moves	3
Current # pebbles	3
Max # pebbles so far	3

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

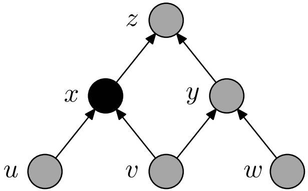


# moves	4
Current # pebbles	2
Max # pebbles so far	3

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them
2. Can always remove black pebble from vertex

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

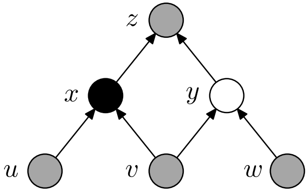


# moves	5
Current # pebbles	1
Max # pebbles so far	3

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them
2. Can always remove black pebble from vertex

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

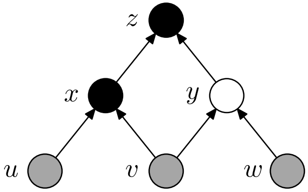


# moves	6
Current # pebbles	2
Max # pebbles so far	3

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them
2. Can always remove black pebble from vertex
3. Can always place white pebble on (empty) vertex

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

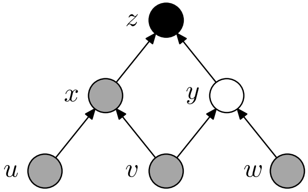


# moves	7
Current # pebbles	3
Max # pebbles so far	3

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them
2. Can always remove black pebble from vertex
3. Can always place white pebble on (empty) vertex

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

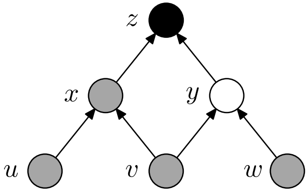


# moves	8
Current # pebbles	2
Max # pebbles so far	3

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them
2. Can always remove black pebble from vertex
3. Can always place white pebble on (empty) vertex

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

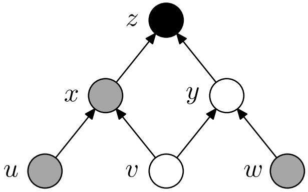


# moves	8
Current # pebbles	2
Max # pebbles so far	3

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them
2. Can always remove black pebble from vertex
3. Can always place white pebble on (empty) vertex
4. Can remove white pebble if all predecessors have pebbles

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

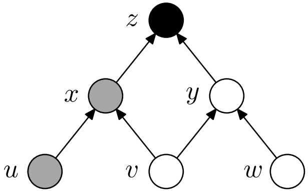


# moves	9
Current # pebbles	3
Max # pebbles so far	3

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them
2. Can always remove black pebble from vertex
3. Can always place white pebble on (empty) vertex
4. Can remove white pebble if all predecessors have pebbles

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

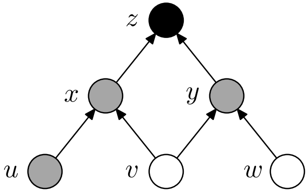


# moves	10
Current # pebbles	4
Max # pebbles so far	4

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them
2. Can always remove black pebble from vertex
3. Can always place white pebble on (empty) vertex
4. Can remove white pebble if all predecessors have pebbles

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

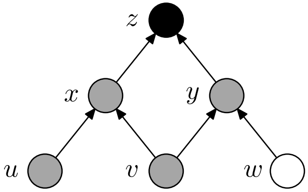


# moves	11
Current # pebbles	3
Max # pebbles so far	4

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them
2. Can always remove black pebble from vertex
3. Can always place white pebble on (empty) vertex
4. Can remove white pebble if all predecessors have pebbles

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G

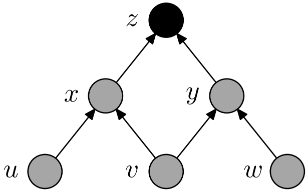


# moves	12
Current # pebbles	2
Max # pebbles so far	4

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them
2. Can always remove black pebble from vertex
3. Can always place white pebble on (empty) vertex
4. Can remove white pebble if all predecessors have pebbles

The Black-White Pebble Game

Goal: get single black pebble on sink vertex z of G



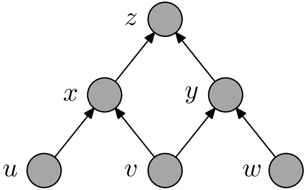
# moves	13
Current # pebbles	1
Max # pebbles so far	4

1. Can place black pebble on (empty) vertex v if all predecessors (vertices with edges to v) have pebbles on them
2. Can always remove black pebble from vertex
3. Can always place white pebble on (empty) vertex
4. Can remove white pebble if all predecessors have pebbles

Use Pebbling Formulas...

CNF formulas encoding so-called pebble games on DAGs

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$

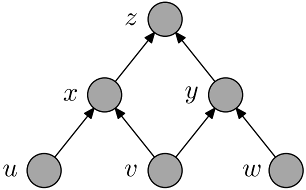


- sources are true
- truth propagates upwards
- but sink is false

Use Pebbling Formulas. . .

CNF formulas encoding so-called pebble games on DAGs

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

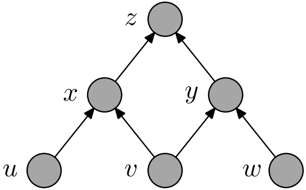


- sources are true
- truth propagates upwards
- but sink is false

Use Pebbling Formulas...

CNF formulas encoding so-called pebble games on DAGs

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$

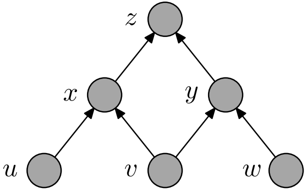


- sources are true
- truth propagates upwards
- but sink is false

Use Pebbling Formulas...

CNF formulas encoding so-called pebble games on DAGs

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$

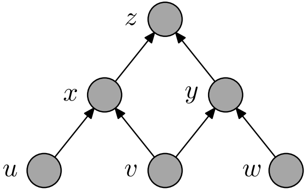


- sources are true
- truth propagates upwards
- but sink is false

Use Pebbling Formulas...

CNF formulas encoding so-called pebble games on DAGs

1. u
2. v
3. w
4. $\overline{u} \vee \overline{v} \vee x$
5. $\overline{v} \vee \overline{w} \vee y$
6. $\overline{x} \vee \overline{y} \vee z$
7. $\overline{z}$



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

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

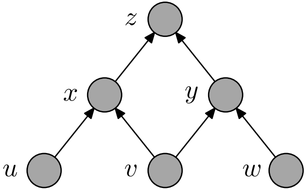
Pebbling formulas studied by [Bonet et al. '98, Raz & McKenzie '99, Ben-Sasson & Wigderson '99] and others

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

Use Pebbling Formulas...

CNF formulas encoding so-called pebble games on DAGs

1. u
2. v
3. w
4. $\overline{u} \lor \overline{v} \lor x$
5. $\overline{v} \lor \overline{w} \lor y$
6. $\overline{x} \lor \overline{y} \lor z$
7. $\overline{z}$



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

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

Pebbling formulas studied by [Bonet et al. '98, Raz & McKenzie '99, Ben-Sasson & Wigderson '99] and others

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

...with Functions Substituted for Variables

Won't work — pebbling formulas solved by unit propagation, so supereasy

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

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

$$\Downarrow$$

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

$$\Downarrow$$

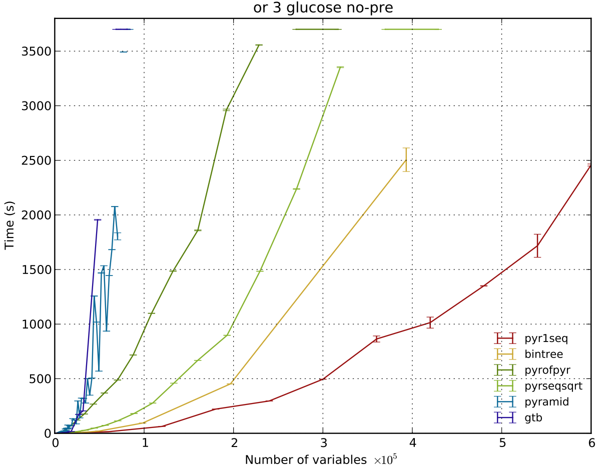
$$\begin{aligned} & (x_1 \vee \overline{x}_2 \vee y_1 \vee y_2) \\ \wedge\ & (x_1 \vee \overline{x}_2 \vee \overline{y}_1 \vee \overline{y}_2) \\ \wedge\ & (\overline{x}_1 \vee x_2 \vee y_1 \vee y_2) \\ \wedge\ & (\overline{x}_1 \vee x_2 \vee \overline{y}_1 \vee \overline{y}_2) \end{aligned}$$

Now CNF formula inherits pebbling graph properties!

About the Experiments

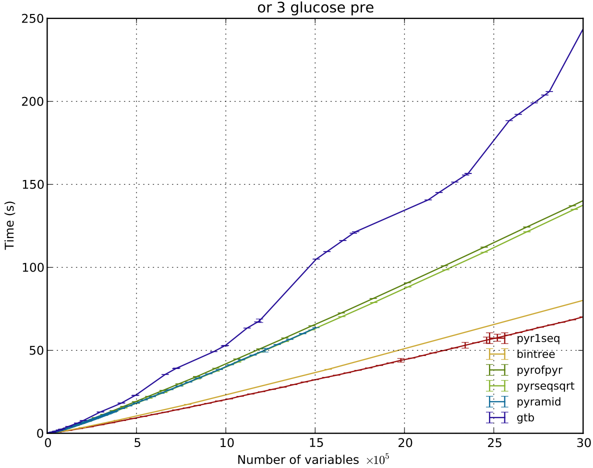
- 12 graph families with varying space complexity
- 11 different functions used to obtain CNF formulas from graphs
- Total of 132 formula families with around 50 instances per family
- CDCL solvers Minisat 2.2, Glucose 2.2, and Lingeling ala
- Experiments
  - with and without preprocessing
  - with and without random shuffling of formulas
- AMD Opteron 2.2 GHz CPU (2374 HE) with 16 GB of memory
- Time-out 1 hour per instance
- Massive amounts of data...

Example Results for Glucose Without Preprocessing



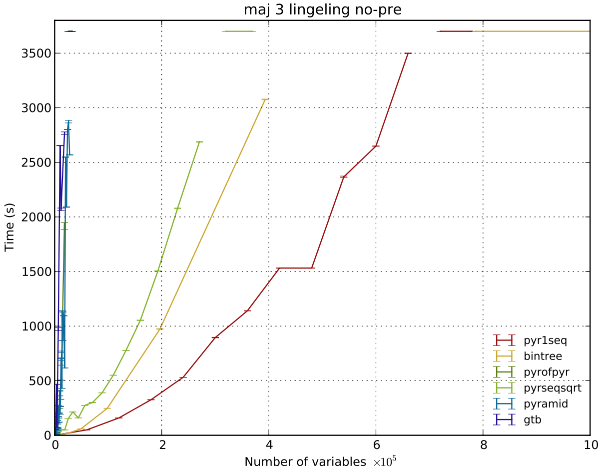
Looks nice. . . "Easy" formulas solved fast; "hard" ones take longer time

Example Results for Glucose with Preprocessing



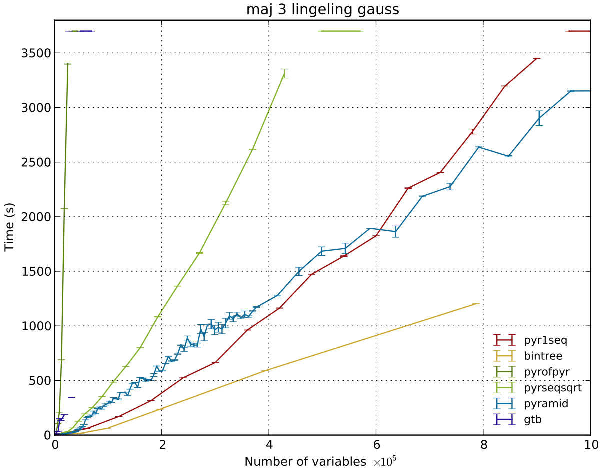
Preprocessing makes formulas much easier, but this still looks nice

Example Results for Lingeling Without Preprocessing



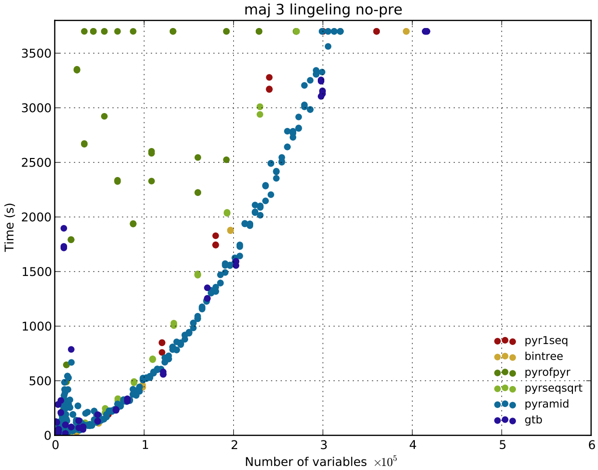
These results still make sense, but are a bit less pretty

Example Results for Lingeling With Preprocessing



Hmmm... Turning on preprocessing, formula families get "out of order"

More Lingeling Results (Without Preprocessing)



And sometimes hard to describe results otherwise than as fairly random. . .

Discussion (1/3): Theory vs. Practice

Dependence on substitution functions

- In theory all functions equal — clearly not the case in practice
- Sometimes we can understand why, but more often not

Discussion (1/3): Theory vs. Practice

Dependence on substitution functions

- In theory all functions equal — clearly not the case in practice
- Sometimes we can understand why, but more often not

Dependence on graph structure

- Graphs with many source vertices yield many binary clauses
- Seems to make formulas easier than space complexity would suggest

Discussion (1/3): Theory vs. Practice

Dependence on substitution functions

- In theory all functions equal — clearly not the case in practice
- Sometimes we can understand why, but more often not

Dependence on graph structure

- Graphs with many source vertices yield many binary clauses
- Seems to make formulas easier than space complexity would suggest

Pebbling space complexity gives limited information

- There is more to pebbling than space complexity
- Sometimes important to pebble graph in exactly the right order
- Corresponding formulas seem harder than their space complexity

Discussion (1/3): Theory vs. Practice

Dependence on substitution functions

- In theory all functions equal — clearly not the case in practice
- Sometimes we can understand why, but more often not

Dependence on graph structure

- Graphs with many source vertices yield many binary clauses
- Seems to make formulas easier than space complexity would suggest

Pebbling space complexity gives limited information

- There is more to pebbling than space complexity
- Sometimes important to pebble graph in exactly the right order
- Corresponding formulas seem harder than their space complexity

The problem of easy benchmarks

- All formulas easy by design — very short proofs in small width
- By design: Want to isolate space complexity as the relevant parameter
- But means SAT solvers can "get lucky"

Discussion (2/3): Behaviour of Different SAT Solvers

MiniSAT and Glucose

- Similar behaviour
- Fairly well-behaved/regular

Discussion (2/3): Behaviour of Different SAT Solvers

MiniSAT and Glucose

- Similar behaviour
- Fairly well-behaved/regular

Lingeling

- Exhibits much “wilder” behaviour in two ways:
  - Less correlation between running time and space complexity
  - Sometimes larger formulas easier than smaller ones from same family

Discussion (2/3): Behaviour of Different SAT Solvers

MiniSAT and Glucose

- Similar behaviour
- Fairly well-behaved/regular

Lingeling

- Exhibits much "wilder" behaviour in two ways:
  - Less correlation between running time and space complexity
  - Sometimes larger formulas easier than smaller ones from same family

Effects of preprocessing

- Always improves running time, but much more significantly for MiniSAT/Glucose (and dampens correlation with space complexity)
- Not surprising — formulas amenable to preprocessing by construction
- Also, space measure doesn't capture what happens during preprocessing

Discussion (3/3): Criticism of Benchmarks

Artificial benchmarks

- True, but the only formulas where we know how to control space
- In general, computing space complexity probably PSPACE-complete
- And computing width complexity provably EXPTIME-complete [Berkholz '12]

Discussion (3/3): Criticism of Benchmarks

Artificial benchmarks

- True, but the only formulas where we know how to control space
- In general, computing space complexity probably PSPACE-complete
- And computing width complexity provably EXPTIME-complete [Berkholz '12]

Varying width and space independently would be more convincing

- Very true, but provably impossible since space $\geq$ width
- What we can hope to determine is whether space is "more fine-grained" hardness indicator

Some Open Problems

- Eliminate most obvious reasons of discrepancy between theory and practice (e.g. # source vertices, order dependence)
- Do theoretical time-space trade-offs turn up in practice for CDCL solvers?
- How does space complexity (and other complexity measures) correlate with running time for algebraic SAT solvers?
- Understand relations of measures such as space and degree better for algebraic solvers (corresponding to polynomial calculus proof system)
- Build better SAT solvers based on algebra or geometry!

Summing up

- Modern CDCL SAT solvers amazingly successful in practice
- But poorly understood which formulas are easy or hard
- We propose space complexity as a measure of hardness in practice
- Don't claim conclusive evidence, but nontrivial correlations
- Believe there are more connections between proof complexity and SAT solving worth exploring

Summing up

- Modern CDCL SAT solvers amazingly successful in practice
- But poorly understood which formulas are easy or hard
- We propose space complexity as a measure of hardness in practice
- Don't claim conclusive evidence, but nontrivial correlations
- Believe there are more connections between proof complexity and SAT solving worth exploring

Advert: We have postdoc and PhD student openings!

Summing up

- Modern CDCL SAT solvers amazingly successful in practice
- But poorly understood which formulas are easy or hard
- We propose space complexity as a measure of hardness in practice
- Don't claim conclusive evidence, but nontrivial correlations
- Believe there are more connections between proof complexity and SAT solving worth exploring

Advert: We have postdoc and PhD student openings!

Thank you for your attention!