# Trustworthy Boolean Reasoning
# 1B: Unsatisfiability Proofs

Randal E. Bryant

Carnegie Mellon University

June, 2022

# Important Ideas for These Lectures

- SAT solvers are useful tools
  - Many practical problems reducible to SAT
  - Need to learn effective encoding techniques
- For many applications, formulas should be unsatisfiable
  - **Program should generate a checkable proof**
  - **There is a well-developed proof infrastructure**
- Binary Decision Diagrams (BDDs) can play important role
  - In supplementing current SAT algorithms
  - In proof generation

# Example Formula

## DIMACS Format

- Standard for all solvers
- Positive integers for variables
- Negative integers for their negations
- Lists terminated with 0

| ID | Clause | DIMACS Encoding |
|---|---|---|
| | | `p cnf 4 6` |
| 1 | $\overline{a} \vee \overline{b} \vee \overline{c}$ | `-1 -2 -3 0` |
| 2 | $\overline{a} \vee \overline{b} \vee c$ | `-1 -2 3 0` |
| 3 | $a \vee \overline{d}$ | `1 -4 0` |
| 4 | $a \vee d$ | `1 4 0` |
| 5 | $b \vee \overline{d}$ | `2 -4 0` |
| 6 | $b \vee d$ | `2 4 0` |

# Example Proof

- Derive empty clause $\bot$ through set of resolution steps

$$\dfrac{\dfrac{\dfrac{\dfrac{a \vee d \qquad a \vee \overline{d}}{a} \qquad \overline{a} \vee \overline{b} \vee c}{\overline{b} \vee c} \qquad \dfrac{a \qquad \overline{a} \vee \overline{b} \vee \overline{c}}{\overline{b} \vee \overline{c}}}{\overline{b}} \qquad \dfrac{b \vee \overline{d} \qquad b \vee d}{b}}{\bot}$$

*But how can a program find such a proof?*

# Unit Propagation

**Unit clauses**

- If formula contains clause $(x)$, then $x$ must be assigned 1.
- If formula contains clause $(\overline{x})$, then $x$ must be assigned 0.

**Propagating Unit Literal $\ell$**

- If $\ell \in C$ and $\ell = 0$, then $C \leftarrow C - \{\ell\}$
- If $\ell \in C$ and $\ell = 1$, then $C$ satisfied
- If any clause becomes unit, then iterate

| Step | Formula | Units |
|---|---|---|
| 1 | $(\boldsymbol{a} \vee \overline{\boldsymbol{c}}) \wedge (\overline{\boldsymbol{a}} \vee \overline{\boldsymbol{b}} \vee \overline{\boldsymbol{c}}) \wedge (\boldsymbol{a} \vee \boldsymbol{b} \vee \boldsymbol{c}) \wedge (\boldsymbol{c})$ | $c = 1$ |
| 2 | $(\boldsymbol{a} \vee \overline{c}) \wedge (\overline{\boldsymbol{a}} \vee \overline{\boldsymbol{b}} \vee \overline{c}) \wedge (\boldsymbol{a} \vee \boldsymbol{b} \vee \boldsymbol{c}) \wedge (\boldsymbol{c})$ | $a = 1$ |
| 3 | $(\boldsymbol{a} \vee \overline{c}) \wedge (\overline{a} \vee \overline{\boldsymbol{b}} \vee \overline{c}) \wedge (\boldsymbol{a} \vee \boldsymbol{b} \vee \boldsymbol{c}) \wedge (\boldsymbol{c})$ | $b = 0$ |
| – | $(\boldsymbol{a} \vee \overline{c}) \wedge (\overline{a} \vee \overline{\boldsymbol{b}} \vee \overline{c}) \wedge (\boldsymbol{a} \vee b \vee \boldsymbol{c}) \wedge (\boldsymbol{c})$ | |

# Basic CDCL Operation

**Conflict-Driven Clause Learning**

- Algorithm in state-of-the art solvers
- Search, but learn from dead ends

```
while(True):
    depth ⟵ 0
    while(True):
        UnitPropagate()
        if all clauses satisfied
            return solution
        if ConflictDetected():
            Generate conflict clause
            break
        Choose variable and assign 0 or 1
        depth ⟵ depth + 1
    if depth = 0:
        return UNSAT
```

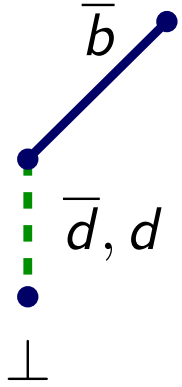
# CDCL Execution Example

- No initial unit propagations

| ID | Clause | UProp? |
|---|---|---|
| 1 | $\overline{a} \vee \overline{b} \vee \overline{c}$ | |
| 2 | $\overline{a} \vee \overline{b} \vee c$ | |
| 3 | $a \vee \overline{d}$ | |
| 4 | $a \vee d$ | |
| 5 | $b \vee \overline{d}$ | |
| 6 | $b \vee d$ | |

# CDCL Execution Example

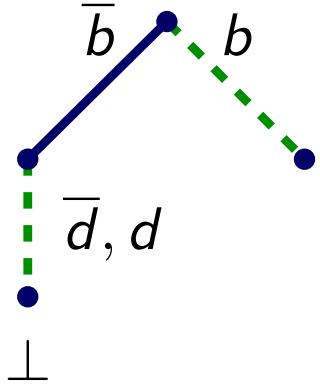
- Setting $b = 0$ causes conflict. Learn clause $b$.




| ID | Clause | UProp? |
|---|---|---|
| 1 | $\overline{a} \vee \overline{b} \vee \overline{c}$ | |
| 2 | $\overline{a} \vee \overline{b} \vee c$ | |
| 3 | $a \vee \overline{d}$ | |
| 4 | $a \vee d$ | |
| 5 | $b \vee \overline{d}$ | * |
| 6 | $b \vee d$ | * |
| 7 | $b$ | |

# CDCL Execution Example

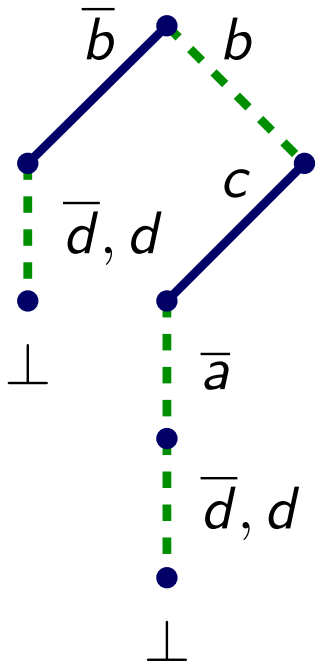
- Unit propagate $b$. No conflict




| ID | Clause | UProp? |
|---|---|---|
| 1 | $\overline{a} \vee \overline{b} \vee \overline{c}$ | |
| 2 | $\overline{a} \vee \overline{b} \vee c$ | |
| 3 | $a \vee \overline{d}$ | |
| 4 | $a \vee d$ | |
| 5 | $b \vee \overline{d}$ | |
| 6 | $b \vee d$ | |
| 7 | $b$ | |

# CDCL Execution Example

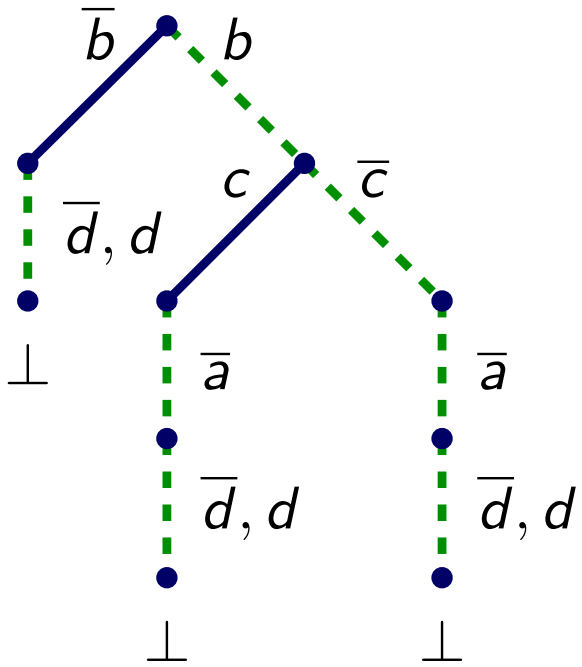
- Setting $c = 1$ causes conflict. Learn clause $\overline{c}$.




| ID | Clause | UProp? |
|---|---|---|
| 1 | $\overline{a} \vee \overline{b} \vee \overline{c}$ | * |
| 2 | $\overline{a} \vee \overline{b} \vee c$ | |
| 3 | $a \vee \overline{d}$ | * |
| 4 | $a \vee d$ | * |
| 5 | $b \vee \overline{d}$ | |
| 6 | $b \vee d$ | |
| 7 | $b$ | * |
| 8 | $\overline{c}$ | |

# CDCL Execution Example

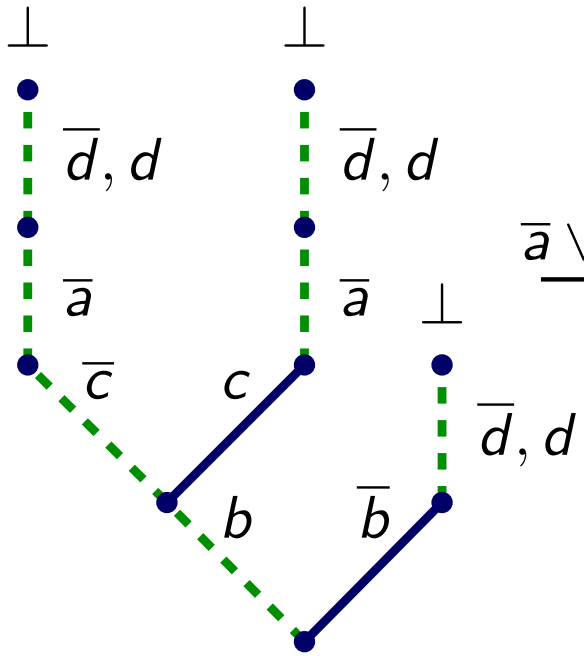
- Unit propagate $b$ and $\overline{c}$. Causes conflict. UNSAT!




| ID | Clause | UProp? |
|---|---|---|
| 1 | $\overline{a} \vee \overline{b} \vee \overline{c}$ | |
| 2 | $\overline{a} \vee \overline{b} \vee c$ | * |
| 3 | $a \vee \overline{d}$ | * |
| 4 | $a \vee d$ | * |
| 5 | $b \vee \overline{d}$ | |
| 6 | $b \vee d$ | |
| 7 | $b$ | * |
| 8 | $\overline{c}$ | * |
| 9 | $\bot$ | |

# Proof from CDCL Run

**Proof Follows Branching Structure of CDCL**




$$\dfrac{\dfrac{\dfrac{\dfrac{a \vee d \quad a \vee \overline{d}}{a} \quad \overline{a} \vee \overline{b} \vee c}{\overline{b} \vee c} \quad \dfrac{a \quad \overline{a} \vee \overline{b} \vee \overline{c}}{\overline{b} \vee \overline{c}}}{\overline{b}} \quad \dfrac{b \vee \overline{d} \quad b \vee d}{b}}{\bot}$$

# Reverse Unit Propagation (RUP)

**Purpose**

- Simple and efficient rule for use by proof checkers
- Good match to operation of CDCL solvers

**Operation**

- Each RUP application forms one step of unsatisfiability proof
- Performs a linear sequence of resolutions steps + subsumption

**Objective**

- $C = \{\ell_1, \ell_2, \ldots, \ell_m\}$ Clause to be added to proof
- $D_1, D_2, \ldots, D_k$ Previous clauses ("Antecedents")
- Prove: $D_1 \wedge D_2 \wedge \cdots \wedge D_k \rightarrow C$

# Reverse Unit Propagation (RUP)

**Objective**

- $C = \{\ell_1, \ell_2, \ldots, \ell_m\}$ Clause to be added to proof
- $D_1, D_2, \ldots, D_k$ Previous clauses ("Antecedents")
- Prove: $D_1 \wedge D_2 \wedge \cdots \wedge D_k \rightarrow C$

**Method**

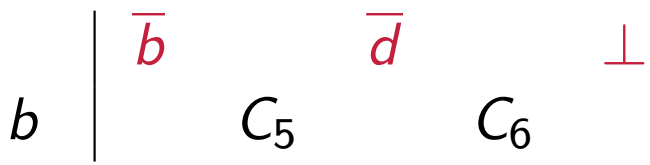
- Assume $\neg C = \overline{\ell}_1 \wedge \overline{\ell}_2 \wedge \cdots \wedge \overline{\ell}_m$
  - $m$ unit clauses!
- Show contradiction with $D_1 \wedge D_2 \wedge \cdots \wedge D_k$
  - Accumulate unit clauses, starting with those for $\neg C$.
  - Accrue more unit clauses from $D_1, D_2, \ldots, D_{k-1}$.
  - When encounter $D_k$, should have contradiction

# RUP Proof Example

| ID | Clause | Antecedents |
|---|---|---|
| $C_1$ | $\overline{a} \vee \overline{b} \vee \overline{c}$ | |
| $C_2$ | $\overline{a} \vee \overline{b} \vee c$ | |
| $C_3$ | $a \vee \overline{d}$ | |
| $C_4$ | $a \vee d$ | |
| $C_5$ | $b \vee \overline{d}$ | |
| $C_6$ | $b \vee d$ | |

# RUP Proof Example

| ID | Clause | Antecedents |
|---|---|---|
| $C_1$ | $\overline{a} \vee \overline{b} \vee \overline{c}$ | |
| $C_2$ | $\overline{a} \vee \overline{b} \vee c$ | |
| $C_3$ | $a \vee \overline{d}$ | |
| $C_4$ | $a \vee d$ | |
| $C_5$ | $b \vee \overline{d}$ | |
| $C_6$ | $b \vee d$ | |
| $C_7$ | $b$ | $C_5, C_6$ |



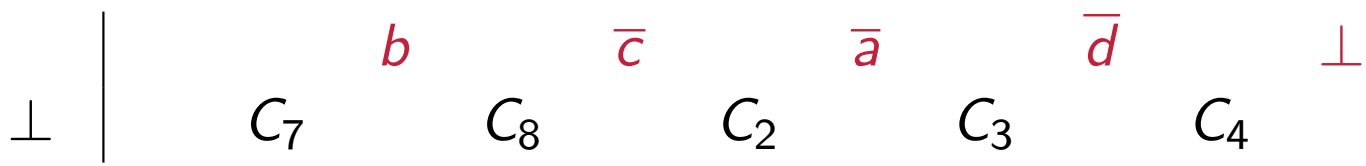
# RUP Proof Example

| ID | Clause | Antecedents |
|---|---|---|
| $C_1$ | $\overline{a} \vee \overline{b} \vee \overline{c}$ | |
| $C_2$ | $\overline{a} \vee \overline{b} \vee c$ | |
| $C_3$ | $a \vee \overline{d}$ | |
| $C_4$ | $a \vee d$ | |
| $C_5$ | $b \vee \overline{d}$ | |
| $C_6$ | $b \vee d$ | |
| $C_7$ | $b$ | $C_5, C_6$ |
| $C_8$ | $\overline{c}$ | $C_7, C_1, C_3, C_4$ |

| | $c$ | $b$ | $\overline{a}$ | $\overline{d}$ | $\bot$ |
|---|---|---|---|---|---|
| $\overline{c}$ | $C_7$ | $C_1$ | $C_3$ | $C_4$ | |

# RUP Proof Example

| ID | Clause | Antecedents |
|---|---|---|
| $C_1$ | $\overline{a} \vee \overline{b} \vee \overline{c}$ | |
| $C_2$ | $\overline{a} \vee \overline{b} \vee c$ | |
| $C_3$ | $a \vee \overline{d}$ | |
| $C_4$ | $a \vee d$ | |
| $C_5$ | $b \vee \overline{d}$ | |
| $C_6$ | $b \vee d$ | |
| $C_7$ | $b$ | $C_5, C_6$ |
| $C_8$ | $\overline{c}$ | $C_7, C_1, C_3, C_4$ |
| $C_9$ | $\bot$ | $C_7, C_8, C_2, C_3, C_4$ |



# Proof File Examples

Proof

| | | |
|---|---|---|
| $C_7$ | $b$ | $C_5, C_6$ |
| $C_8$ | $\overline{c}$ | $C_7, C_1, C_3, C_4$ |
| $C_9$ | $\perp$ | $C_7, C_8, C_2, C_3, C_4$ |

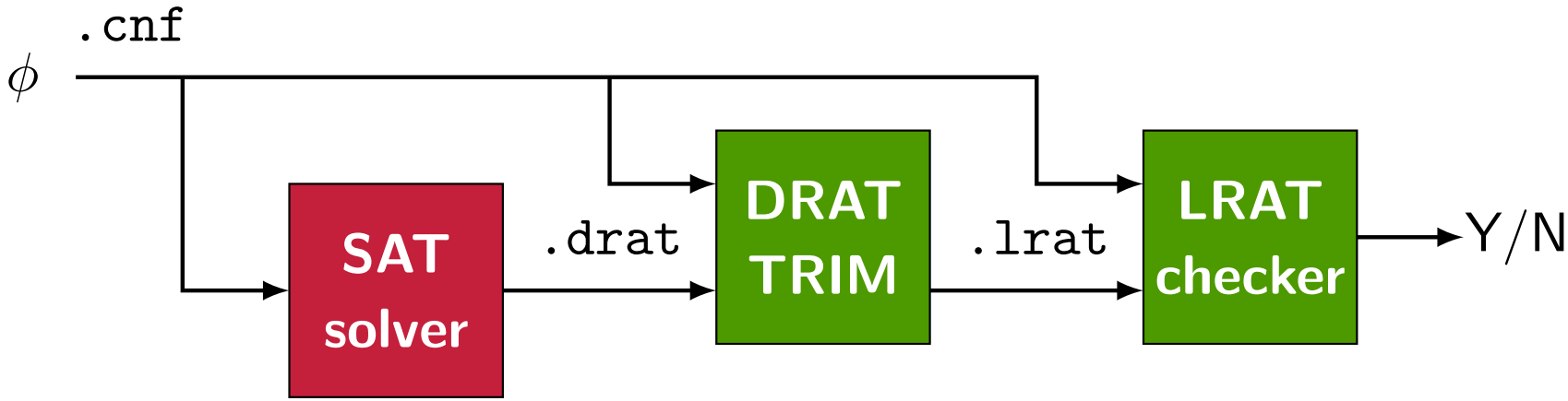
DRAT Proof File

```
2 0
-3 0
0
```

LRAT Proof File

```
7    2 0     5 6 0
8    -3 0    7 1 3 4 0
9    0       7 8 2 3 4 0
```

# Proof Checking Infrastructure




**Operation:**

- DRAT-TRIM adds antecedents to proof steps
- LRAT checker simply checks each proof step

**LRAT Checkers:**

- LRAT-CHECK written in C.
  - Fast and high capacity
  - Designed to be simple enough to easily understand
- Formally verified ones. Built on ACL2, Coq, HOL, . . .
  - Integrity not compromised if solver or DRAT-TRIM has bug

# Resolution and CDCL

**CDCL $\approx$ Resolution**

- *Strength:* CDCL solver can readily generate resolution proofs
- *Weakness:* Lower bound on performance

**Example: Pigeonhole Principle(PHP)**

- Problem:
  - $n$ holes, $n+1$ pigeons
  - Assign pigeons to holes:
    - Each pigeon is assigned to some hole
    - Each hole has at most one pigeon
- SAT Encoding:
  - Variables: $p_{i,j}$: Pigeon $j$ in hole $i$. $1 \leq i \leq n$, $1 \leq j \leq n+1$.
  - $n+1$ at-least-one constraints
  - $n$ at-most-one constraints
  - $O(n^3)$ total clauses
- PHP($n$) resolution proofs are exponential in $n$ [Haken, 1985]

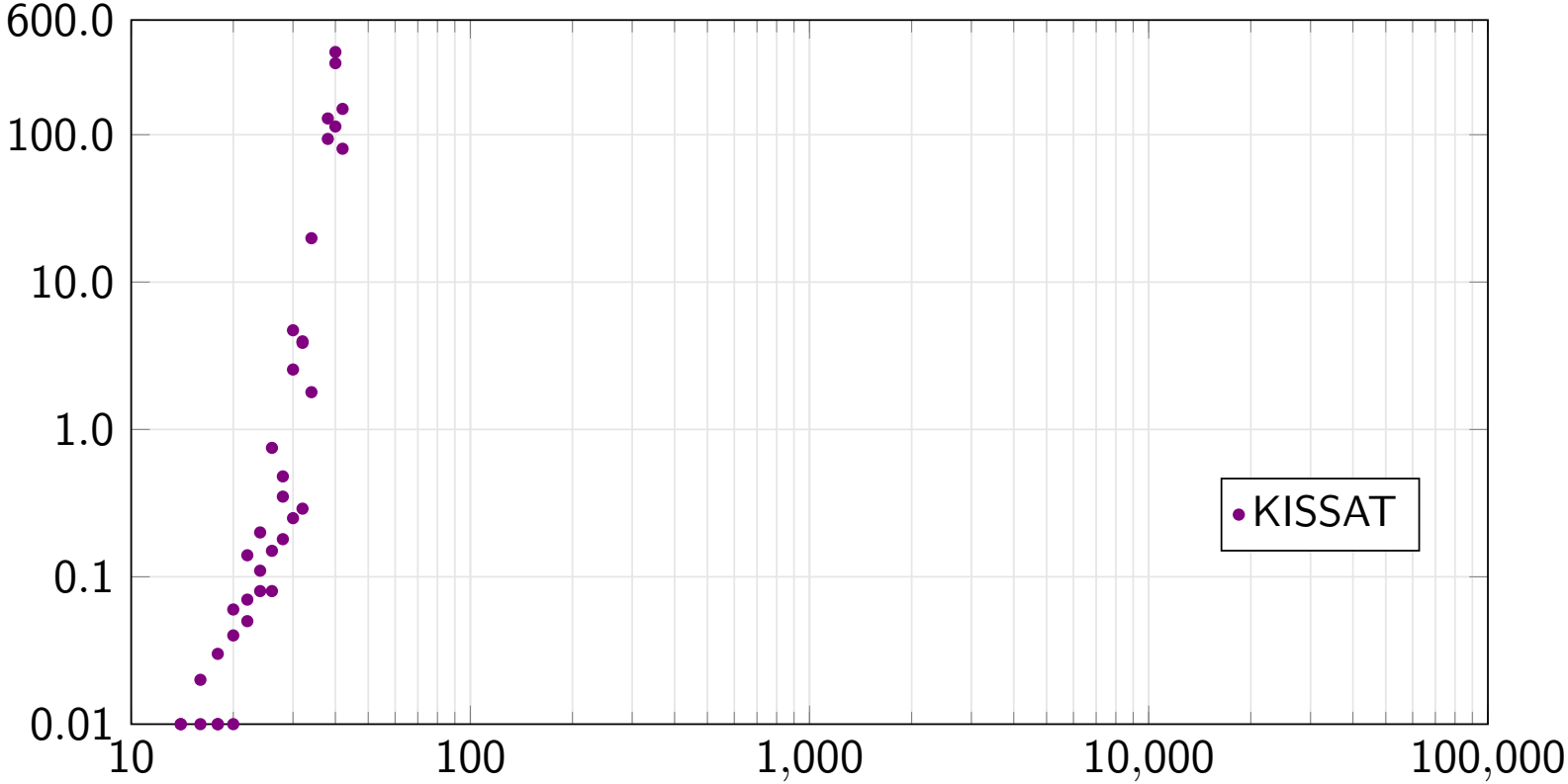
# Parity Benchmark

- Chew and Heule, SAT 2020
- For random permtuation $\pi$:

$$\begin{array}{lllllllll} x_1 & \oplus & x_2 & \oplus & \cdots & \oplus & x_n & = & 1 \quad \text{Odd parity} \\ x_{\pi(1)} & \oplus & x_{\pi(2)} & \oplus & \cdots & \oplus & x_{\pi(n)} & = & 0 \quad \text{Even parity} \end{array}$$

- Conjunction unsatisfiable
- Very challenging for CDCL solvers
- Unit propagation of limited value
  - $k$-way parity constraint
  - Only propagate when $k-1$ variables assigned

# Parity Benchmark Runtime




- KISSAT: State-of-the-art CDCL solver
- Tried 3-different seeds for each value of $n$
- Limited to $n \leq 42$ within 600 seconds

# Extended Resolution

- Tseitin, 1967

**Can introduce extension variables**

- Variable $z$ that has not yet occurred in proof
- Must add *defining* clauses
  - Encode constraint of form $z \leftrightarrow F$
  - Boolean formula $z$ over input and earlier extension variables

**Extension variable $z$ becomes shorthand for formula $F$**

- Repeated use can yield exponentially smaller proof

**Similar to use of encoding variables in SAT formulas**

- That's why they're called "Tseitin variables"
- But here they become part of proof, not of input formula

# Extended RUP Proof Example

| ID | Clause | Antecedents |
|---|---|---|
| $C_1$ | $\overline{a} \vee \overline{b} \vee \overline{c}$ | |
| $C_2$ | $\overline{a} \vee \overline{b} \vee c$ | |
| $C_3$ | $a \vee \overline{d}$ | |
| $C_4$ | $a \vee d$ | |
| $C_5$ | $b \vee \overline{d}$ | |
| $C_6$ | $b \vee d$ | |

**Strategy**

- Use $z$ to encode $a \wedge b$.
- E.g., $C_1$ becomes $\overline{z} \vee \overline{c}$.

# Extended RUP Proof Example

| ID | Clause | Antecedents |
|---|---|---|
| $C_1$ | $\overline{a} \vee \overline{b} \vee \overline{c}$ | |
| $C_2$ | $\overline{a} \vee \overline{b} \vee c$ | |
| $C_3$ | $a \vee \overline{d}$ | |
| $C_4$ | $a \vee d$ | |
| $C_5$ | $b \vee \overline{d}$ | |
| $C_6$ | $b \vee d$ | |
| $C_7$ | $\overline{z} \vee a$ | Defining Clauses |
| $C_8$ | $\overline{z} \vee b$ | |
| $C_9$ | $z \vee \overline{a} \vee \overline{b}$ | |

# Extended RUP Proof Example

| ID | Clause | Antecedents |
|---|---|---|
| $C_1$ | $\overline{a} \vee \overline{b} \vee \overline{c}$ | |
| $C_2$ | $\overline{a} \vee \overline{b} \vee c$ | |
| $C_3$ | $a \vee \overline{d}$ | |
| $C_4$ | $a \vee d$ | |
| $C_5$ | $b \vee \overline{d}$ | |
| $C_6$ | $b \vee d$ | |
| $C_7$ | $\overline{z} \vee a$ | Defining Clauses |
| $C_8$ | $\overline{z} \vee b$ | |
| $C_9$ | $z \vee \overline{a} \vee \overline{b}$ | |
| $C_{10}$ | $\overline{z} \vee \overline{c}$ | $C_7, C_8, C_1$ |
| $C_{11}$ | $\overline{z}$ | $C_7, C_8, C_2, C_{10}$ |
| $C_{12}$ | $d$ | $C_4, C_6, C_9, C_{11}$ |
| $C_{13}$ | $\bot$ | $C_{12}, C_3, C_5, C_9, C_{11}$ |

# Can Extended Resolution Yield Faster SAT Solvers?

**PHP Proof Complexity**

- Exponential for ordinary resolution
- $O(n^4)$ for extended resolution [Cook, 1976]

**Use in SAT?**

- No clear way to choose which formulas to abbreviate
- No clear way to shorten search by using abbreviations