

# Seminar decision procedures: Certification of SAT and unSAT proofs

Wolfgang Nicka

Technische Universität München

June 14, 2016

# Boolean satisfiability problem

## Term

The **boolean satisfiability problem** (*SAT*) asks whether there exists an interpretation of a boolean formula such that the formula evaluates to true.

## Example

$$(\bar{x} \wedge y \vee a) \wedge (x \wedge b) \wedge (\bar{a})$$

*SAT* is NP-complete

# Basics

**Literal** Named boolean variable

**Clause** Disjunction of literals

**Conjunctive normal form** Conjunction of clauses

**Satisfiable** A formula is satisfiable if all clauses are satisfiable at the same time

From here on out all boolean formulae are assumed to be in conjunctive normal form (CNF).

# SAT solvers

Different SAT solvers use a variety of strategies

Although experimentally verified solvers may contain bugs.

## Example

*Lingeling: evaluated by millions of benchmarks, bug discovered **over one and a half years after** the industry started using it. [WHHJ14]*

⇒ Proof traces and certification

# Proof Traces

**Proof traces** are a sequence of clauses that are redundant with respect to a formula.

## Example

$$(a \vee b \vee c) \wedge (a \vee \bar{d})$$

Added clauses called **lemmas**.

# Proof Traces

**Proof traces** are a sequence of clauses that are redundant with respect to a formula.

## Example

$$(a \vee b \vee c) \wedge (a \vee \bar{d}) \wedge (a \vee \bar{a})$$

Added clauses called **lemmas**.

# CDCL proofs

Most state-of-the-art SAT solvers, called **conflict-driven clause learning (CDCL)** solvers, prove unsatisfiability by adding lemmas.

Idea:

One **proof trace checker** to verify output from multiple SAT solvers.

⇒ Need for a **common proof format**.

# Proof formats

## Term

A **proof format** expresses how to check whether a clause added by a CDCL solver is redundant with respect to the formula.

**Resolution-based proof formats:** simple validation, but proofs are large and the corresponding checkers have high complexity.

**Clausal proof formats:** proofs are compact and easy to emit, but verification is slower.

Because resolution based proofs are large and it is harder to modify current SAT solvers to emit them we focus on clausal proof formats.



# Resolvents

## Term

If a CNF formula  $F$  contains clauses  $A = x \vee \bigvee a_i$  and  $B = \bar{x} \vee \bigvee b_i$ , then  $C = \bigvee a_i \vee \bigvee b_i$  is logically implied.  $C$  is called the **resolvent** of  $A$  and  $B$ .

## Example

$$(a \vee b \vee c) \wedge (b \vee \bar{d}) \wedge (\bar{c} \vee e \vee \bar{f})$$

# Resolvents

## Term

If a CNF formula  $F$  contains clauses  $A = x \vee \bigvee a_i$  and  $B = \bar{x} \vee \bigvee b_i$ , then  $C = \bigvee a_i \vee \bigvee b_i$  is logically implied.  $C$  is called the **resolvent** of  $A$  and  $B$ .

## Example

$$(a \vee b \vee c) \wedge (b \vee \bar{d}) \wedge (\bar{c} \vee e \vee \bar{f}) \wedge (a \vee b \vee e \vee \bar{f})$$

# Boolean constraint propagation

## Term

**Boolean constraint propagation** or **unit propagation** is the procedure of fixing literals to true or false based on unit clauses, i.e. clauses with only one literal.

## Example

$$F = (a) \wedge (\bar{a} \vee \bar{b}) \wedge (\bar{a} \vee b) \wedge (\bar{b} \vee c)$$

# Boolean constraint propagation

## Term

**Boolean constraint propagation** or **unit propagation** is the procedure of fixing literals to true or false based on unit clauses, i.e. clauses with only one literal.

## Example

$$F = (a) \wedge (\bar{a} \vee \bar{b}) \wedge (\bar{a} \vee b) \wedge (\bar{b} \vee c) \equiv a \wedge \bar{b} \wedge b \wedge (\bar{b} \vee c)$$

# Boolean constraint propagation

## Term

**Boolean constraint propagation** or **unit propagation** is the procedure of fixing literals to true or false based on unit clauses, i.e. clauses with only one literal.

## Example

$$\begin{aligned} F &= (a) \wedge (\bar{a} \vee \bar{b}) \wedge (\bar{a} \vee b) \wedge (\bar{b} \vee c) \equiv a \wedge \bar{b} \wedge b \wedge (\bar{b} \vee c) \\ &\equiv a \wedge \bar{b} \wedge \emptyset. \end{aligned}$$

# Boolean constraint propagation

## Term

**Boolean constraint propagation** or **unit propagation** is the procedure of fixing literals to true or false based on unit clauses, i.e. clauses with only one literal.

## Example

$$\begin{aligned} F &= (a) \wedge (\bar{a} \vee \bar{b}) \wedge (\bar{a} \vee b) \wedge (\bar{b} \vee c) \equiv a \wedge \bar{b} \wedge b \wedge (\bar{b} \vee c) \\ &\equiv a \wedge \bar{b} \wedge \emptyset. \end{aligned}$$

## Definition

The simplified formula resulting from fixing a unit clause from  $F$  to true is referred to as **BCP(F)**.

# Redundancy properties

## Term

**Asymmetric literal addition (ALA)** for a clause  $C$  in  $F$  adds redundant literals to  $C$  based on other clauses that otherwise consist of only literals in  $C$ .

## Example

$$C = (a \vee b \vee c \vee \bar{e}), F = D = (a \vee c \vee d)$$

# Redundancy properties

## Term

**Asymmetric literal addition (ALA)** for a clause  $C$  in  $F$  adds redundant literals to  $C$  based on other clauses that otherwise consist of only literals in  $C$ .

## Example

$$C = (a \vee b \vee c \vee \bar{e}), F = D = (a \vee c \vee d)$$

$$\implies ALA(F, C) = (a \vee b \vee c \vee \bar{e} \vee \bar{d})$$



# Redundancy properties

## Term

**Asymmetric literal addition (ALA)** for a clause  $C$  in  $F$  adds redundant literals to  $C$  based on other clauses that otherwise consist of only literals in  $C$ .

## Example

$$C = (a \vee b \vee c \vee \bar{e}), F = D = (a \vee c \vee d) \\ \implies ALA(F, C) = (a \vee b \vee c \vee \bar{e} \vee \bar{d})$$

## Definition

A clause  $C$  has **asymmetric tautology (AT)** with respect to  $F$  iff  $ALA(F, C)$  is a tautology (has T). A clause  $C$  with AT is also called a **reverse unit propagation (RUP)** clause.

# RAT redundancy property

## Definition

A clause  $C$  in a CNF formula  $F$  has the property **resolution asymmetric tautology (RAT)** if either  $C$  has AT or all resolvents of  $C$  and clauses in  $F$  on one fixed literal in  $C$  have AT.

## Example

$$(\bar{a} \vee b) \wedge (c \vee b \vee d) \wedge (b \vee \bar{d})$$

# RAT redundancy property

## Definition

A clause  $C$  in a CNF formula  $F$  has the property **resolution asymmetric tautology (RAT)** if either  $C$  has AT or all resolvents of  $C$  and clauses in  $F$  on one fixed literal in  $C$  have AT.

## Example

$$(\bar{a} \vee b) \wedge (c \vee b \vee d) \wedge (b \vee \bar{d}) \wedge (a \vee c)$$

# RAT redundancy property

## Definition

A clause  $C$  in a CNF formula  $F$  has the property **resolution asymmetric tautology (RAT)** if either  $C$  has AT or all resolvents of  $C$  and clauses in  $F$  on one fixed literal in  $C$  have AT.

## Example

$$(\bar{a} \vee b) \wedge (c \vee b \vee d) \wedge (b \vee \bar{d}) \wedge (a \vee c)$$

*All resolvents on  $a$ :  $(c \vee b)$*

# RAT redundancy property

## Definition

A clause  $C$  in a CNF formula  $F$  has the property **resolution asymmetric tautology (RAT)** if either  $C$  has AT or all resolvents of  $C$  and clauses in  $F$  on one fixed literal in  $C$  have AT.

## Example

$$(\bar{a} \vee b) \wedge (c \vee b \vee d) \wedge (b \vee \bar{d}) \wedge (a \vee c)$$

All resolvents on  $a$ :  $(c \vee b)$

$ALA(F, (c \vee b)) = (c \vee b \vee \bar{d} \vee d)$  has  $T$ , so  $(c \vee b)$  has AT.

# RAT redundancy property

## Definition

A clause  $C$  in a CNF formula  $F$  has the property **resolution asymmetric tautology (RAT)** if either  $C$  has AT or all resolvents of  $C$  and clauses in  $F$  on one fixed literal in  $C$  have AT.

## Example

$$(\bar{a} \vee b) \wedge (c \vee b \vee d) \wedge (b \vee \bar{d}) \wedge (a \vee c)$$

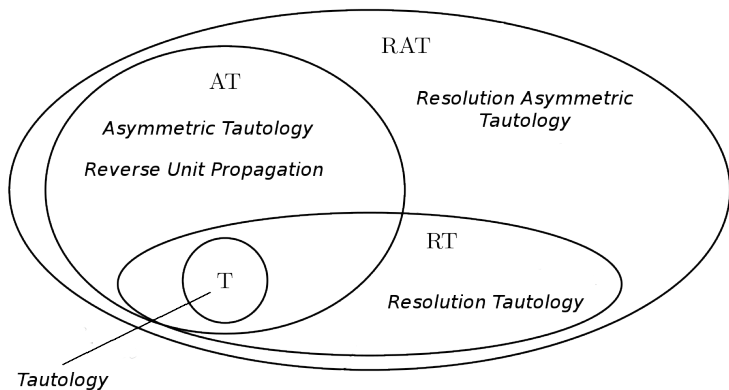
All resolvents on  $a$ :  $(c \vee b)$

$ALA(F, (c \vee b)) = (c \vee b \vee \bar{d} \vee d)$  has  $T$ , so  $(c \vee b)$  has AT.

AT preserves **logical equivalence**.

RAT preserves **satisfiability equivalence**.

# Overview



[HHJW13b]

# Reverse Unit Propagation Checks

A lemma  $L$  is RUP / has AT with respect to  $F$  and all previously added lemmas iff unit propagation on its negation results in a conflict.

## Example

$$(b \vee c \vee d) \wedge (b \vee \bar{d}), C = (b \vee c)$$



# Reverse Unit Propagation Checks

A lemma  $L$  is RUP / has AT with respect to  $F$  and all previously added lemmas iff unit propagation on its negation results in a conflict.

## Example

$$\frac{(b \vee c \vee d) \wedge (b \vee \bar{d}), C = (b \vee c)}{(b \vee c) = (\bar{b}) \wedge (\bar{c})}$$

# Reverse Unit Propagation Checks

A lemma  $L$  is RUP / has AT with respect to  $F$  and all previously added lemmas iff unit propagation on its negation results in a conflict.

## Example

$$(b \vee c \vee d) \wedge (b \vee \bar{d}), C = (b \vee c)$$

$$\overline{(b \vee c)} = (\bar{b}) \wedge (\bar{c})$$

$$BCP((b \vee c \vee d) \wedge (b \vee \bar{d}) \wedge (\bar{b}) \wedge (\bar{c})) = (d) \wedge \emptyset \wedge (\bar{b}) \wedge (\bar{c})$$

# Reverse Unit Propagation Checks

A lemma  $L$  is RUP / has AT with respect to  $F$  and all previously added lemmas iff unit propagation on its negation results in a conflict.

## Example

$$(b \vee c \vee d) \wedge (b \vee \bar{d}), C = (b \vee c)$$

$$\overline{(b \vee c)} = (\bar{b}) \wedge (\bar{c})$$

$$BCP((b \vee c \vee d) \wedge (b \vee \bar{d}) \wedge (\bar{b}) \wedge (\bar{c})) = (d) \wedge \emptyset \wedge (\bar{b}) \wedge (\bar{c})$$

$\Rightarrow$  RUP proof checker [HHJW13b]

## Checking clausal proofs

It is also possible to save **clause deletion information** along with addition of lemmas.

⇒ DRUP proof format [HHJW13a]

For the more inclusive RAT proof format:

The lemma  $L$  has RAT with respect to  $F$  if it has AT or has RAT on its first literal.

⇒ RAT proof format [HHJW13b]

Combining RUP checks, deletion information and RAT checks:

⇒ DRAT proof format [WHHJ14]

# Examples - DIMACS CNF and DRAT

CNF formula

```
p cnf 4 10
 1  2 -3  0
-1 -2  3  0
-1 -2 -3  0
 2  3 -4  0
-2 -3  4  0
-1 -3 -4  0
 1  3  4  0
-1  2  4  0
 1 -2 -4  0
-1  2 -4  0
```

DRAT proof

```
-1  0
d -1  2  4  0
  2  0
  0
```

# Conclusion




The tool "DRAT-trim":

- ▶ accepts RUP and DRUP proofs
- ▶ DRAT proof traces are easy to emit
- ▶ checks proofs using additional techniques, like bounded variable addition

[WHHJ14]

**THANK YOU**

# Bibliography I

-  Marijn Heule, Warren A Hunt Jr, and Nathan Wetzler, *Trimming while checking clausal proofs.*, FMCAD, 2013, pp. 181–188.
-  Marijn JH Heule, Warren A Hunt Jr, and Nathan Wetzler, *Verifying refutations with extended resolution*, Automated Deduction–CADE-24, Springer, 2013, pp. 345–359.
-  Nathan Wetzler, Marijn JH Heule, and Warren A Hunt Jr, *Drat-trim: Efficient checking and trimming using expressive clausal proofs*, Theory and Applications of Satisfiability Testing–SAT 2014, Springer, 2014, pp. 422–429.