

# Expressing Symmetry Breaking in DRAT Proofs

Marijn J.H. Heule



*Joint work with*  
Warren Hunt, Jr. and Nathan Wetzler

CADE-25, August 7, 2015

Introduction and Motivation

Symmetry Breaking in SAT Solvers

Breaking a Single Symmetry

Breaking Multiple Symmetries

Tools and Evaluation

Conclusions

# Motivation

Satisfiability solvers are used in amazing ways...

- ▶ Hardware verification: Centaur x86 verification
- ▶ Combinatorial problems:
  - ▶ Ramsey numbers and van der Waerden numbers  
[Dransfield, Marek, and Truszczynski, 2004; Kouril and Paul, 2008]
  - ▶ Gardens of Eden in Conway's Game of Life  
[Hartman, Heule, Kwekkeboom, and Noels, 2013]
  - ▶ Erdős Discrepancy Problem [Konev and Lisitsa, 2014]

# Motivation

Satisfiability solvers are used in amazing ways...

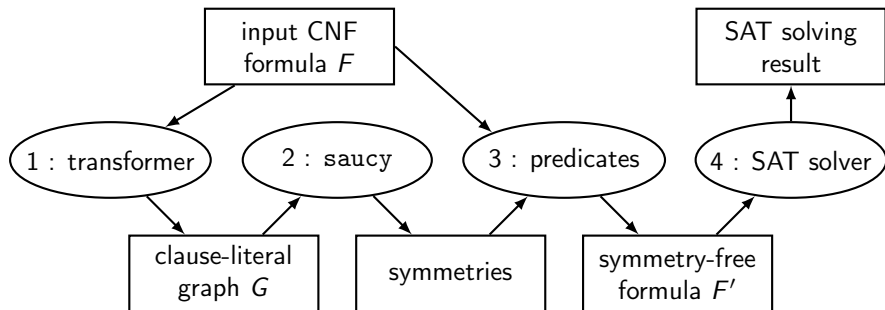
- ▶ Hardware verification: Centaur x86 verification
- ▶ Combinatorial problems:
  - ▶ Ramsey numbers and van der Waerden numbers  
[Dransfield, Marek, and Truszczynski, 2004; Kouril and Paul, 2008]
  - ▶ Gardens of Eden in Conway's Game of Life  
[Hartman, Heule, Kwekkeboom, and Noels, 2013]
  - ▶ Erdős Discrepancy Problem [Konev and Lisitsa, 2014]

..., but satisfiability solvers have errors.

- ▶ Documented bugs in SAT, SMT, and QBF solvers  
[Brummayer and Biere, 2009; Brummayer et al., 2010]
- ▶ Implementation errors often imply conceptual errors
- ▶ Symmetry breaking, which is crucial to solve combinatorial problems, cannot be validated with existing methods

# Symmetry Breaking Tool Chain

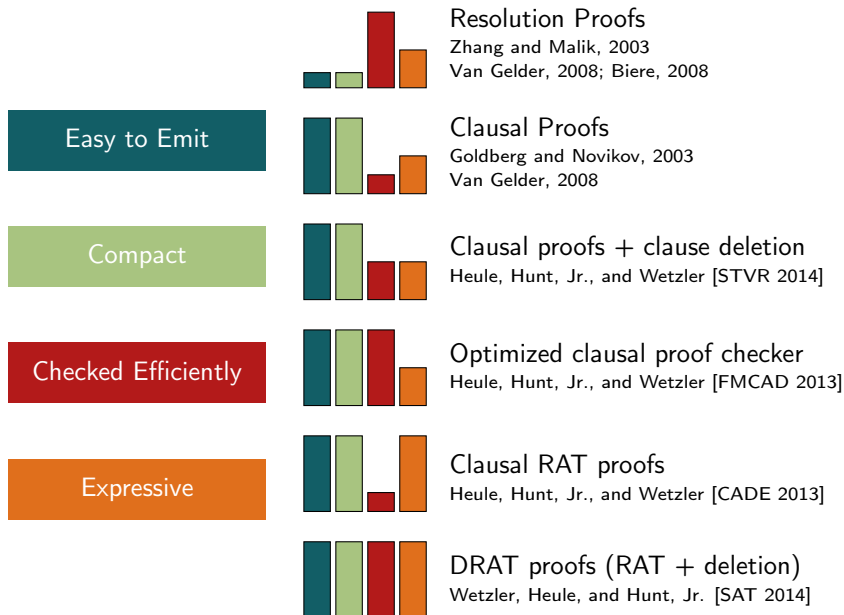
1. The input formula is transformed into a **clause-literal graph**;
2. A symmetry detection tool **extracts symmetries** from the graph;
3. **Symmetry-breaking predicates** are added to the input formula;
4. The symmetry-free formula is solved using a **SAT solver**.



A bug in any of these tools may result in **incorrect results**

Most **observed bugs** during SAT Competition 2013 were caused by tools 1-3

# From Resolution to Clausal DRAT Proofs



## Main Contribution

We present a method to express the addition of symmetry-breaking predicates in DRAT, a clausal proof format supported by top-tier solvers.

Our method allows, for the first time, validation of SAT solver results obtained via symmetry breaking, thereby validating the results of symmetry extraction tools as well.

# Symmetry Breaking in SAT Solvers



## Example Formulas: Unavoidable Subgraphs

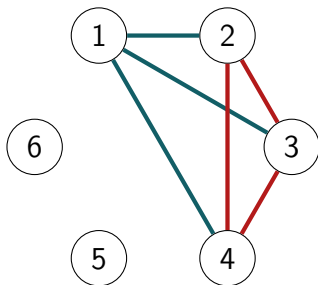
A connected undirected graph  $G$  is an **unavoidable subgraph** of clique  $K$  of order  $n$  if **any red/blue edge-coloring** of the edges of  $K$  contains  $G$  either in red or in blue.

**Ramsey Number  $R(k)$ :** What is the smallest  $n$  such that any graph with  $n$  vertices has either a clique or a co-clique of size  $k$ ?

$$R(3) = 6$$

$$R(4) = 18$$

$$43 \leq R(5) \leq 49$$



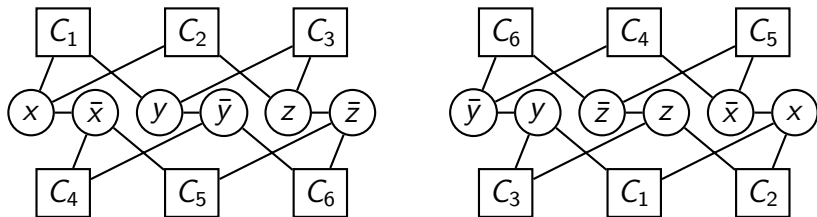
SAT solvers can determine that  $R(4) = 18$  in **1 second** using symmetry breaking; w/o symmetry breaking it requires **weeks**.

## Example formula: an unavoidable path of two edges

Consider the formula below — which expresses the statement whether path of two edges unavoidable in a clique of order 3:

$$F := \underbrace{(x \vee y)}_{C_1} \wedge \underbrace{(x \vee z)}_{C_2} \wedge \underbrace{(y \vee z)}_{C_3} \wedge \underbrace{(\bar{x} \vee \bar{y})}_{C_4} \wedge \underbrace{(\bar{x} \vee \bar{z})}_{C_5} \wedge \underbrace{(\bar{y} \vee \bar{z})}_{C_6}$$

A **clause-literal graph** has a vertex for each clause and literal, and edges for each literal occurrence connecting the literal and clause vertex. Also, two complementary literals are connected.



Symmetry:  $(x, y, z)(\bar{y}, \bar{z}, \bar{x})$  is an **edge-preserving bijection**

## Convert Symmetries into Symmetry-Breaking Predicates

A **symmetry**  $\sigma = (x_1, \dots, x_n)(p_1, \dots, p_n)$  of a CNF formula  $F$  is an edge-preserving bijection of the clause-literal graph of  $F$ , that maps literals  $x_i$  onto  $p_i$  and  $\bar{x}_i$  onto  $\bar{p}_i$  with  $i \in \{1..n\}$ .

Given a CNF formula  $F$ . Let  $\tau$  be a satisfying truth assignment for  $F$  and  $\sigma$  a symmetry for  $F$ , then  $\sigma(\tau)$  is also a satisfying truth assignment for  $F$ .

Symmetry  $\sigma = (x_1, \dots, x_n)(p_1, \dots, p_n)$  for  $F$  can be broken by adding a **symmetry-breaking predicate**:  $x_1, \dots, x_n \leq p_1, \dots, p_n$ .

$$\begin{aligned} &(\bar{x}_1 \vee p_1) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee p_2) \wedge (p_1 \vee \bar{x}_2 \vee p_2) \wedge \\ &(\bar{x}_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee p_3) \wedge (\bar{x}_1 \vee p_2 \vee \bar{x}_3 \vee p_3) \wedge \\ &(p_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee p_3) \wedge (p_1 \vee p_2 \vee \bar{x}_3 \vee p_3) \wedge \dots \end{aligned}$$

Why are we allowed to add these clauses?

# Breaking a Single Symmetry

Given a clause  $C = (l_1 \vee \dots \vee l_k)$  and a CNF formula  $F$ :

- ▶  $\bar{C}$  denotes the conjunction of its negated literals  $(\bar{l}_1) \wedge \dots \wedge (\bar{l}_k)$
- ▶  $F \vdash_1 \epsilon$  denotes that unit propagation on  $F$  derives a conflict
- ▶  $C$  is an **asymmetric tautology** w.r.t.  $F$  if and only if  $F \wedge \bar{C} \vdash_1 \epsilon$
- ▶  $C$  is a **resolution asymmetric tautology** on  $l \in C$  w.r.t.  $F$  iff for all resolvents  $C \diamond D$  with  $D \in F$  and  $\bar{l} \in D$  holds that  $F \wedge \overline{C \diamond D} \vdash_1 \epsilon$

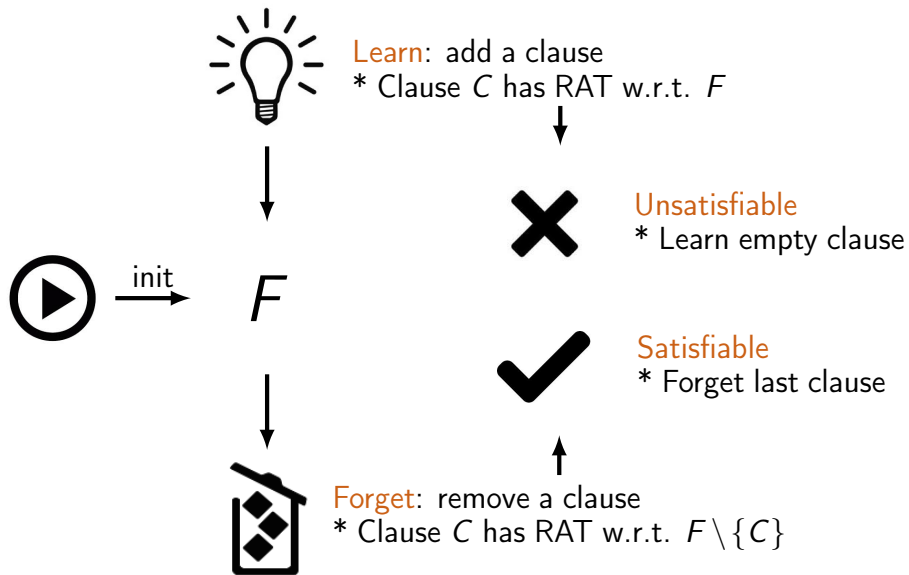
## Example

Consider the formula  $F = (a \vee c) \wedge (\bar{b} \vee \bar{c}) \wedge (b \vee d)$ :

- ▶ The clause  $(a \vee d)$  is an asymmetric tautology w.r.t.  $F$
- ▶ The clause  $(b \vee c)$  is a resolution asymmetric tautology w.r.t.  $F$

Theorem: Given a formula  $F$  and a clause  $C$  having **RAT** with respect to  $F$ , then  $F$  and  $F \cup \{C\}$  are **equi-satisfiable**.

# Clausal Proof System using RAT addition and deletion



## Expressing a Symmetry Breaking Predicate in DRAT (1)

Introduce auxiliary variables using  $\sigma = (x_1, \dots, x_n)(p_1, \dots, p_n)$

- ▶ The **swap variable**  $s := x_1, \dots, x_n > p_1, \dots, p_n$
- ▶ The **prime variables**  $x'_i := \begin{cases} p_i & \text{if } s \text{ set to true} \\ x_i & \text{otherwise} \end{cases}$

Example (using  $\sigma = (x_1, x_2)(x_2, x_1)$ )

- ▶  $add(s \vee \bar{x}_1 \vee x_2), add(\bar{s} \vee x_1), add(\bar{s} \vee \bar{x}_2)$
- ▶  $add(x'_1 \vee \bar{s} \vee \bar{x}_2), add(\bar{x}'_1 \vee \bar{s} \vee x_2), add(x'_1 \vee s \vee \bar{x}_1),$   
 $add(\bar{x}'_1 \vee s \vee x_1)$

Add symmetry-breaking predicate using the prime variables:

- ▶ Add the **constraint**  $x'_1, \dots, x'_n \leq p'_1, \dots, p'_n$

Example (using  $\sigma = (x_1, x_2)(x_2, x_1)$ )

- ▶  $add(s \vee \bar{x}'_1 \vee x'_2), add(\bar{x}'_1 \vee x'_2), delete(s \vee \bar{x}'_1 \vee x'_2)$

## Expressing a Symmetry Breaking Predicate in DRAT (2)

Redefine involved clauses

- ▶ For each clause  $C \in F$  that contains at least one literal  $l$  which occurs in the symmetry, add a clause  $C'$  which is a copy of  $C$  with literals  $l'$  for each such  $l$ .
- ▶ Remove the original involved clauses.

Example (using  $\sigma = (x_1, x_2)(x_2, x_1)$  and  $C = (x_2 \vee \bar{x}_3)$ )

- ▶  $add(s \vee x'_2 \vee \bar{x}_3)$ ,  $add(x'_2 \vee \bar{x}_3)$ ,  $delete(s \vee x'_2 \vee \bar{x}_3)$ ,  
 $delete(x_2 \vee \bar{x}_3)$

Optionally remove all definitions of the first step

- ▶ After this step, the resulting formula is equal to the original formula extended with the symmetry-breaking predicate (modulo variable renaming).
- ▶ This step reduces validation costs significantly.



# Breaking Multiple Symmetries

## Difficulties due to Multiple Symmetries

Consider a formula  $F$  with two symmetries

$\sigma_1 = (x_1, \dots, x_n)(p_1, \dots, p_n)$  and

$\sigma_2 = (y_1, \dots, y_n)(q_1, \dots, q_n)$ .

Symmetry breaking will add the predicates

$x_1, \dots, x_n \leq p_1, \dots, p_n$  and  $y_1, \dots, y_n \leq q_1, \dots, q_n$ . The number of predicates is **linear in the number of symmetries**.

After breaking  $\sigma_1$  with the method shown, resulting in formula  $F'$ , we cannot simply apply it again, because  **$\sigma_2$  is not a symmetry of  $F'$** .

Hence, the method shown can only **redefine original clauses**.

To obtain a symmetry-free formula, the method shown has to be applied multiple times per symmetry. Even with just two symmetries, one may need to apply it twice per symmetry.

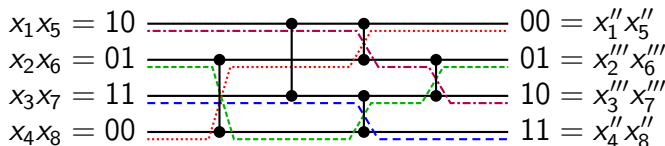
## Break a Symmetry Chain using Sorting Networks

A **symmetry chain** is a sequence of  $k$  symmetries of length  $2n$  with the property that  $x_{i,j} = p_{i,j+n}$ ,  $p_{i,j} = x_{i,j+n}$ , and  $x_{i+1,j} = x_{i,j+n}$  with  $1 \leq i < k$  and  $1 \leq j \leq n$ .

Example (symmetry chain  $(x_1, x_5)(x_2, x_6)(x_3, x_7)(x_4, x_8)$ )

- ▶ based on  $\sigma_i = (x_i, x_{i+4}, x_{i+1}, x_{i+5})(x_{i+1}, x_{i+5}, x_i, x_{i+4})$
- ▶ results in predicates  $x_1, x_5 \leq x_2, x_6 \leq x_3, x_7 \leq x_4, x_8$

Breaking a symmetry chain comes down at sorting the assignments, which can be realized using a sorting network.



Size  $k$  symmetry chain: apply the procedure  $\mathcal{O}(k \log^2 k)$  times.

## Converting Symmetries into a Symmetry Chain

Q: How to break multiple symmetries in general?

A: Convert them into a symmetry chain.

+: Limits the size of the partial proof.

-: Breaks the symmetries only partially.

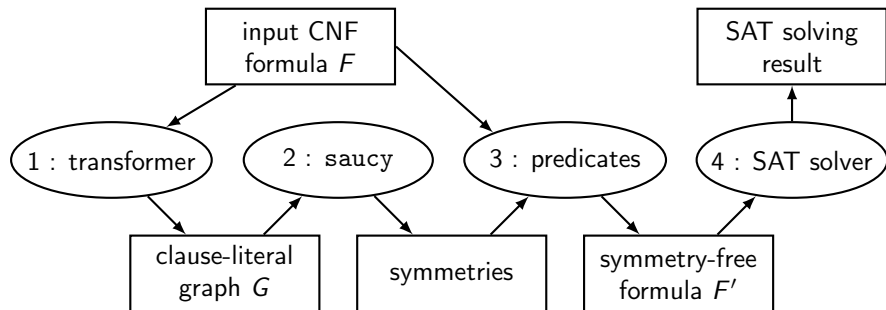
### Example

Consider two symmetries:  $\sigma_1 = (x_1, x_4, x_2, x_5)(x_2, x_5, x_1, x_4)$  and  $\sigma_2 = (x_2, x_4, x_3, x_6)(x_3, x_6, x_2, x_4)$ . Compute reduced symmetries  $\sigma'_1 = (x_1, x_2)(x_2, x_1)$  and  $\sigma'_2 = (x_2, x_3)(x_3, x_2)$  that form a symmetry chain. Using  $\sigma'_1$  and  $\sigma'_2$  to define the swap variable and the symmetry-breaking predicate. Use  $\sigma_1$  and  $\sigma_2$  for the other definitions.

# Tools and Evaluation

# Old Tool Chain

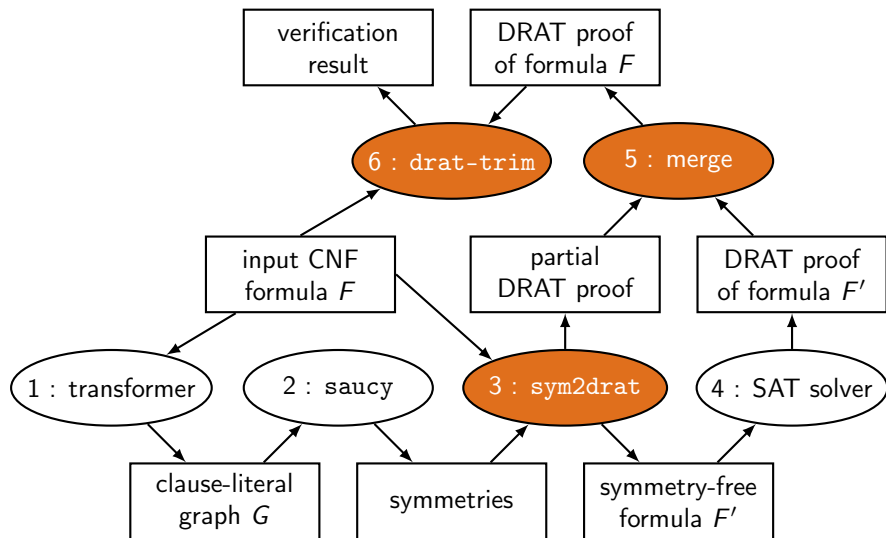
1. The input formula is transformed into a **clause-literal graph**;
2. A symmetry detection tool **extracts symmetries** from the graph;
3. **Symmetry-breaking predicates** are added to the input formula;
4. The symmetry-free formula is solved using a **SAT solver**.



A bug in any of these tools may result in **incorrect results**

Most **observed bugs** during SAT Competition 2013 were caused by tools 1-3

## New Tool Chain



Only the correctness of the proof checker needs to be trusted

## New Tools

The new tool `sym2drat`:

- ▶ Input: CNF formula and symmetries;
- ▶ Output: A symmetry-free formula and a partial proof that describes the derivation from the input formula to the symmetry-free formula;
- ▶ Uses pairwise sorting to reduce the size of the partial proof.

Merge: simply concatenate using Unix `cat`

DRAT proof checkers:

- ▶ Implemented an extension for `drat-trim` to validate partial DRAT proofs. This feature was crucial during development for debugging purposes;
- ▶ Modified our mechanically-verified proof checker to make it compatible with DRAT proofs.



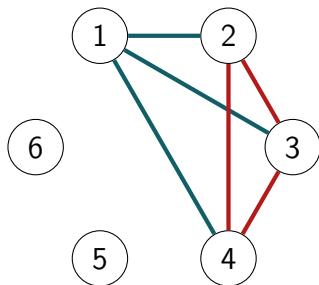
## Evaluation: Ramsey Number Four

**Ramsey Number  $R(k)$ :** What is the smallest  $n$  such that any graph with  $n$  vertices has either a clique or a co-clique of size  $k$ ?

$$R(3) = 6$$

$$R(4) = 18$$

$$43 \leq R(5) \leq 49$$

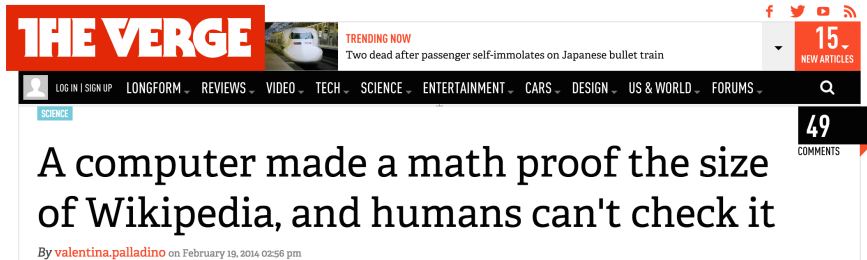


SAT solvers can determine that  $R(4) = 18$  in **1 second** using symmetry breaking; w/o symmetry breaking it requires **weeks**.

The size of the proof is 20 MB and the time required is 1.8 s

Proof validated with our mechanically-verified checker as well.

# Evaluation: Erdős Discrepancy Conjecture



The screenshot shows the top portion of a news article on The Verge website. The site's logo is in the top left. A navigation bar contains links for 'LONGFORM', 'REVIEWS', 'VIDEO', 'TECH', 'SCIENCE', 'ENTERTAINMENT', 'CARS', 'DESIGN', 'US & WORLD', and 'FORUMS'. A 'TRENDING NOW' section features a thumbnail of a train and the headline 'Two dead after passenger self-immolates on Japanese bullet train'. On the right, there are social media icons and a red box indicating '15 NEW ARTICLES'. Below the navigation bar, a 'SCIENCE' tag is visible. The main headline reads 'A computer made a math proof the size of Wikipedia, and humans can't check it'. Below the headline, it says 'By [valentina.palladino](#) on February 19, 2014 02:56 pm'. A '49 COMMENTS' badge is located on the right side of the article header.

Erdős Discrepancy Conjecture was recently solved using SAT. The conjecture states that there exists no infinite sequence of  $-1, +1$  such that for all  $d, k$  holds that  $(x_i \in \{-1, +1\})$ :

$$\left| \sum_{i=1}^k x_{id} \right| \leq 2$$

The original DRAT proof was **13Gb**. Our new proof using symmetry breaking is **2Gb**.

## Evaluation: Two Pigeons per Hole Problems

Biere proposed benchmarks expressing whether  $2n + 1$  pigeons can be put in  $n$  holes that contain at most two pigeons per hole.

For  $n > 6$  they can only be solved by symmetry breaking or cardinality resolution.



No SAT solvers can produce a proof for the problems with  $n > 6$ .

Our method can produce proofs for problems with  $n \leq 12$  that can be generated in minutes and validated within an hour.

# Conclusions

# Conclusions

## Conclusions:

- ▶ The first approach to validate symmetry-breaking techniques usage in SAT solvers by expressing the techniques as DRAT proof steps;
- ▶ Increases the trust in results based on symmetry breaking;
- ▶ Evaluated our method on hard-combinatorial formulas.

## Future work:

- ▶ Determine precisely the number of times the symmetry-breaking procedure needs to be applied;
- ▶ Improve the speed of the mechanically-verified checker;
- ▶ Implement a parallel proof checker to reduce the gap between solving and verification costs.

# Conclusions

## Conclusions:

- ▶ The first approach to validate symmetry-breaking techniques usage in SAT solvers by expressing the techniques as DRAT proof steps;
- ▶ Increases the trust in results based on symmetry breaking;
- ▶ Evaluated our method on hard-combinatorial formulas.

## Future work:

- ▶ Determine precisely the number of times the symmetry-breaking procedure needs to be applied;
- ▶ Improve the speed of the mechanically-verified checker;
- ▶ Implement a parallel proof checker to reduce the gap between solving and verification costs.

Thanks! Questions?