Proof Logging for Preprocessing/Presolving in MaxSAT and 0-1 Integer Linear Programming

Andy Oertel
Lund University & University of Copenhagen

WHOOPS '25

September 13, 2025

Based on work together with Jeremias Berg, Ambros Gleixner, Alexander Hoen, Hannes Ihalainen, Matti Järvisalo, Magnus O. Myreen, Jakob Nordström, and Yong Kiam Tan

0-1 Integer Linear Programming (0-1 ILP)

$$\begin{aligned} \text{min} \quad & 2x_2 + 3x_3 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 \geq 2 \\ & x_2 + x_3 + x_4 \geq 2 \\ & x_1 - 2x_2 - 2x_3 + x_4 \geq 0 \end{aligned}$$

- Specialization of mixed integer programming (MIP)
- Input: 0-1 integer linear program (or pseudo-Boolean formula)
  - Integer linear objective function and collection of integer linear inequalities/constraints
  - Variables with domain $\{0, 1\}$
- Output:
  - Optimal value of objective subject to satisfying all inequalities

Maximum Satisfiability (MaxSAT)

$$\begin{aligned} \min \quad & 2x_2 + 3x_3 \\ \text{s.t.} \quad & x_1 \vee x_2 \vee x_3 \\ & x_2 \vee x_3 \vee x_4 \\ & x_1 \vee \overline{x}_2 \vee \overline{x}_3 \vee x_4 \end{aligned}$$

→ MaxSAT Solver → Result: optimal value 2

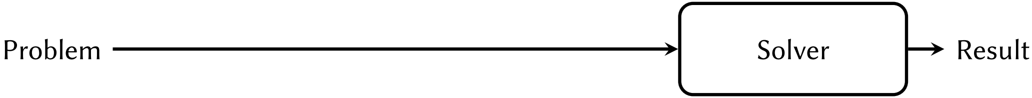
- Optimization variant of SAT problem
- Input: MaxSAT problem
  - Integer linear objective function and collection of clauses
  - Variables with domain $\{0, 1\}$
- Output:
  - Optimal value of objective subject to satisfying all clauses

Maximum Satisfiability (MaxSAT)

$$\begin{aligned} \min \quad & 2x_2 + 3x_3 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 \geq 1 \\ & x_2 + x_3 + x_4 \geq 1 \\ & x_1 + \overline{x}_2 + \overline{x}_3 + x_4 \geq 1 \end{aligned}$$

- Optimization variant of SAT problem
- Input: MaxSAT problem
  - Integer linear objective function and collection of clauses
  - Variables with domain $\{0, 1\}$
- Output:
  - Optimal value of objective subject to satisfying all clauses
- Specialization of 0-1 ILP

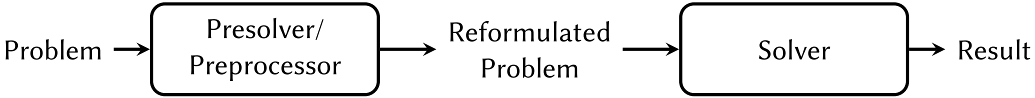
Idea of Presolving/Preprocessing



So far:

- Problem directly given to solver

Idea of Presolving/Preprocessing



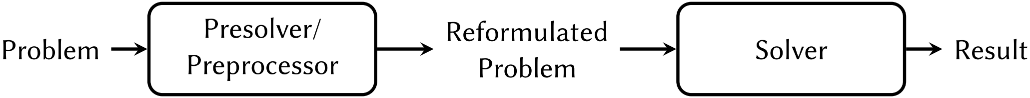
So far:

- Problem directly given to solver

Typical workflow:

- Problem reformulated before it is given to core solver
- Known as *presolving* in the MIP community
- Known as *preprocessing* in the MaxSAT community
- Can be tightly integrated with solver or independent tool

Idea of Presolving/Preprocessing



So far:

- Problem directly given to solver

Typical workflow:

- Problem reformulated before it is given to core solver
- Known as *presolving* in the MIP community
- Known as *preprocessing* in the MaxSAT community
- Can be tightly integrated with solver or independent tool

Important for state-of-the-art performance!

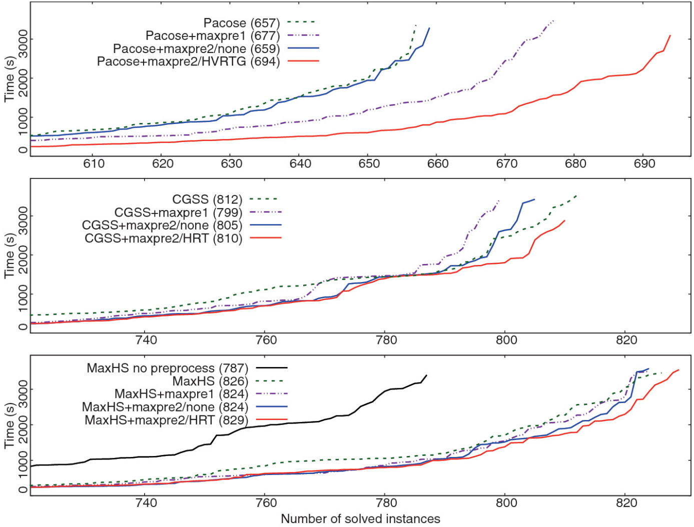
Importance of Presolving in MIP

- Performance analysis of presolve reductions in MIP [ABG+20]

		default	disabled presolving			
bracket	#instances	#timeout	#timeout	#faster	#slower	×slower
all	3047	547	1035	255	1755	3.36
$\geq$ 0 sec	2511	16	504	255	1755	4.52
$\geq$ 1 sec	1944	16	504	210	1634	6.60
$\geq$ 10 sec	1575	16	504	141	1380	9.05
$\geq$ 100 sec	1099	16	504	86	983	12.36
$\geq$ 1000 sec	692	16	504	34	643	19.48

Presolving is one of the most important techniques in mixed-integer programming!

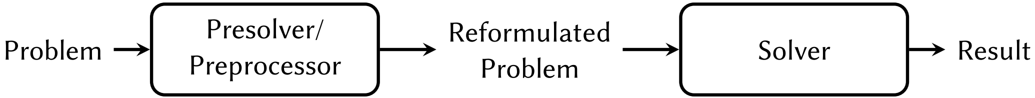
Importance of Preprocessing in MaxSAT



- Performance analysis for MaxSAT preprocessing with MaxPre [IBJ22]

Preprocessing improves performance significantly for many MaxSAT solvers!

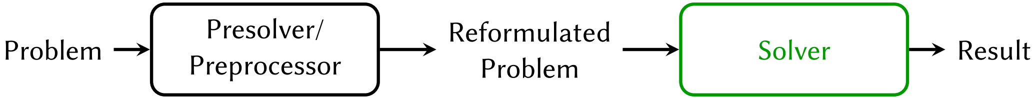
Proof Logging for Workflow Using Problem Reformulation



Goal:

- Certification for whole solving workflow

Proof Logging for Workflow Using Problem Reformulation



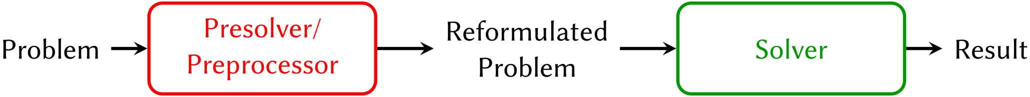
Goal:

- Certification for whole solving workflow

Good news:

- Certification for some MIP solving algorithms using VIPR [CGS17]
- Certification for MaxSAT using VeriPB [VDB22, Van23, BBN+23, BBN+24]

Proof Logging for Workflow Using Problem Reformulation



Goal:

- Certification for whole solving workflow

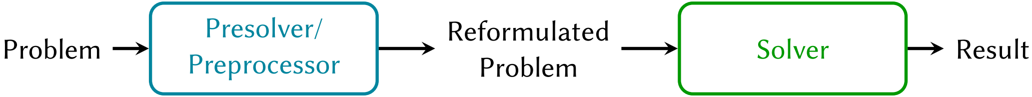
Good news:

- Certification for some MIP solving algorithms using VIPR [CGS17]
- Certification for MaxSAT using VeriPB [VDB22, Van23, BBN+23, BBN+24]

Problem:

- Also need to certify problem reformulations in presolver/preprocessor

Proof Logging for Workflow Using Problem Reformulation



Goal:

- Certification for whole solving workflow

Good news:

- Certification for some MIP solving algorithms using VIPR [CGS17]
- Certification for MaxSAT using VeriPB [VDB22, Van23, BBN$^+$23, BBN$^+$24]

Problem:

- Also need to certify problem reformulations in presolver/preprocessor

This talk:

- Certification of 0-1 ILP presolving and MaxSAT preprocessing
- Formally verified end-to-end verification framework for problem reformulations

Outline

Preliminaries

- **Boolean variable x:** with domain 0 (false) and 1 (true)
- **Literal ℓ:** x or negation $\overline{x} = 1 - x$
- **Pseudo-Boolean (PB) constraint:** integer linear inequality over literals

$$3x_1 + 2x_2 + 5\overline{x}_3 \geq 5$$

- 0-1 integer linear constraint is same as PB constraint
- **Equality constraint:** syntactic sugar for 2 inequalities

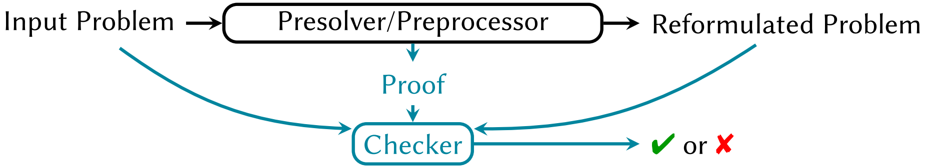
$$3x_1 + 2x_2 + 5\overline{x}_3 = 5 \longrightarrow \begin{array}{l} 3x_1 + 2x_2 + 5\overline{x}_3 \geq 5 \\ 3x_1 + 2x_2 + 5\overline{x}_3 \leq 5 \end{array}$$

- **Clause:** disjunction of literals / at-least-one constraint

$$x_1 \vee \overline{x}_2 \vee \overline{x}_3 \iff x_1 + \overline{x}_2 + \overline{x}_3 \geq 1$$

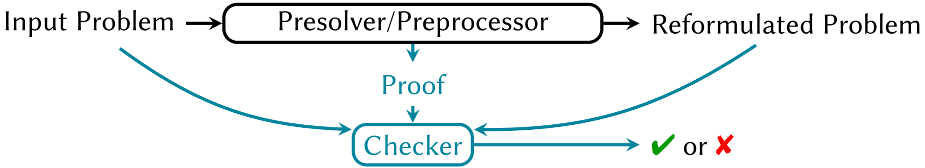
- MaxSAT is special case of 0-1 ILP

Proof Invariants



- Step-by-step modify optimization problem preserving optimal value

Proof Invariants

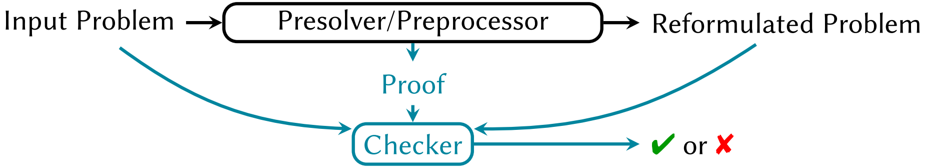


- Step-by-step modify optimization problem preserving optimal value

Two sets of constraints needed:

- Core set $\mathcal{C}$ guarantee:
  - Current problem ($\min f'$, s.t. $\mathcal{C}$) has same optimal value as input problem ($\min f$, s.t. F)
- Derived set $\mathcal{D}$ of constraints are all constraints derived by rules
  - Any solution to $\mathcal{C}$ can be extended to a solution of $\mathcal{C} \cup \mathcal{D}$
- Constraints can be moved from derived to core set

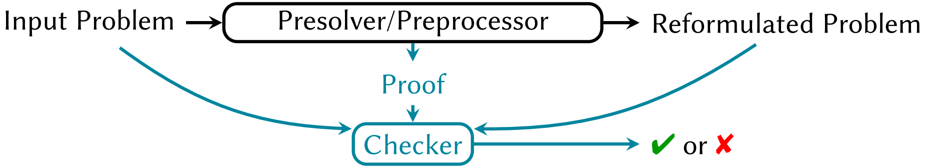
Certifying Problem Reformulations



How to certify presolving/preprocessing?

- Represent each reformulation using proof steps
- Soundness of proof system guarantees that optimal value does not change
- Check that core set and objective at end of proof match output problem

Certifying Problem Reformulations



How to certify presolving/preprocessing?

- Represent each reformulation using proof steps
- Soundness of proof system guarantees that optimal value does not change
- Check that core set and objective at end of proof match output problem

Guarantee:

- Input problem has same optimal value as output problem of presolver/preprocessor

Cutting Planes Proof System [CCT87]

Rules that preserve set of solutions:

- Literal axiom

$$\text{Literal } x \; \frac{}{x \geq 0} \qquad \text{Literal } \overline{x} \; \frac{}{\overline{x} \geq 0}$$

- Addition

$$\text{Addition} \; \frac{x_1 + 2\overline{x}_2 + 2\overline{x}_3 \geq 3 \qquad \overline{x}_2 + 3x_3 \geq 3}{x_1 + 3\overline{x}_2 + x_3 \geq 4}$$

- Multiplication

$$\text{Multiply by 2} \; \frac{x_1 + 2\overline{x}_2 \geq 3}{2x_1 + 4\overline{x}_2 \geq 6}$$

- Division (and rounding up)

$$\text{Divide by 2} \; \frac{2x_1 + 2\overline{x}_2 + 4x_3 \geq 5}{x_1 + \overline{x}_2 + 2x_3 \geq \lceil 2.5 \rceil}$$

Cutting Planes Proof System [CCT87]

Rules that preserve set of solutions:

- Literal axiom

$$\text{Literal } x \ \frac{}{x \geq 0} \qquad \text{Literal } \bar{x} \ \frac{}{\bar{x} \geq 0}$$

- Addition

$$\text{Addition} \ \frac{x_1 + 2\bar{x}_2 + 2\bar{x}_3 \geq 3 \qquad \bar{x}_2 + 3x_3 \geq 3}{x_1 + 3\bar{x}_2 + x_3 \geq 4}$$

- Multiplication

$$\text{Multiply by 2} \ \frac{x_1 + 2\bar{x}_2 \geq 3}{2x_1 + 4\bar{x}_2 \geq 6}$$

- Division (and rounding up)

$$\text{Divide by 2} \ \frac{2x_1 + 2\bar{x}_2 + 4x_3 \geq 5}{x_1 + \bar{x}_2 + 2x_3 \geq 3}$$

Cutting Planes: Example

$$\begin{aligned} \min \quad & x_1 + x_2 \\ \text{s.t.} \quad & 3x_1 + 2x_2 + 3\overline{x}_3 \geq 3 \quad (42) \end{aligned} \longrightarrow \begin{aligned} \min \quad & x_1 + x_2 \\ \text{s.t.} \quad & 3x_1 + 2x_2 + 3\overline{x}_3 \geq 3 \quad (42) \\ & x_1 + \overline{x}_3 \geq 1 \quad (43) \end{aligned}$$

Proof tree:

$$\dfrac{\dfrac{\dfrac{}{\overline{x}_2 \geq 0}\ \text{Literal } \overline{x}_2}{2\overline{x}_2 \geq 0}\ \text{Multiply by 2} \qquad 3x_1 + 2x_2 + 3\overline{x}_3 \geq 3}{\dfrac{3x_1 + 3\overline{x}_3 \geq 1}{x_1 + \overline{x}_3 \geq 1}\ \text{Divide by 3}}\ \text{Addition}$$

VeriPB syntax:

```
pol ~x2 2 * 42 + 3 d ;
```

Redundance-Based Strengthening

- Cutting planes rules preserve set of solutions
- Also want to add or remove solutions

Redundance-Based Strengthening

- Cutting planes rules preserve set of solutions
- Also want to add or remove solutions

Redundance-based strengthening [BT19, GN21]

- Requires substitution ω (mapping variables to truth values or literals)
- We can introduce C with respect to constraints $\mathcal{C} \cup \mathcal{D}$ and objective f if

$$\mathcal{C} \cup \mathcal{D} \cup \{\neg C\} \vdash \mathcal{C} \cup \mathcal{D} \cup C\restriction_\omega \cup \{f \geq f\restriction_\omega\}$$

- ω has to be given explicitly
- Implication should be efficiently checkable:
  - Obvious to proof checker
  - Or explicitly by cutting planes proof

Redundance-Based Strengthening: Example

$$\begin{aligned} \min \quad & x_1 + x_2 \\ \text{s.t.} \quad & x_1 + 2x_2 + 3\overline{x}_3 \geq 3 \qquad (42) \end{aligned} \longrightarrow \begin{aligned} \min \quad & x_1 + x_2 \\ \text{s.t.} \quad & x_1 + 2x_2 + 3\overline{x}_3 \geq 3 \qquad (42) \\ & \overline{x}_3 \geq 1 \qquad (43) \end{aligned}$$

VeriPB syntax:

```
red 1 ~x3 >= 1 : x3 -> 0 ;
```

- $\overline{x}_3 \geq 1$ derived by redundance-based strengthening with $\{\overline{x}_3 \mapsto 0\}$

Redundance-Based Strengthening: Example

$$\begin{aligned} \min \quad & x_1 + x_2 \\ \text{s.t.} \quad & x_1 + 2x_2 + 3\bar{x}_3 \geq 3 \quad (42) \end{aligned} \longrightarrow \begin{aligned} \min \quad & x_1 + x_2 \\ \text{s.t.} \quad & x_1 + 2x_2 + 3\bar{x}_3 \geq 3 \quad (42) \\ & \bar{x}_3 \geq 1 \quad (43) \end{aligned}$$

VeriPB syntax:

```
red 1 ~x3 >= 1 : x3 -> 0 ;
```

- $\bar{x}_3 \geq 1$ derived by redundance-based strengthening with $\{\bar{x}_3 \mapsto 0\}$

$$\mathcal{C} \cup \mathcal{D} \cup \{\neg C\} \vdash (\mathcal{C} \cup \mathcal{D} \cup C)\restriction_\omega \cup \{f \geq f\restriction_\omega\}$$

- All implications are trivial:
  - For constraint (42), $(x_1 + 2x_2 + 3\bar{x}_3 \geq 3)\restriction_\omega$ is $x_1 + 2x_2 \geq 0$
  - For derived constraint, $(\bar{x}_3 \geq 1)\restriction_\omega$ is $0 \geq 0$
  - For objective condition $f \geq f\restriction_\omega$, $x_1 + x_2 \geq (x_1 + x_2)\restriction_\omega$ is $x_1 + x_2 \geq x_1 + x_2$

Usage of Strengthening Rules

Strengthening useful for:

- Basic symmetry breaking
- Without-loss-of-generality reasoning
- Introducing extension variables

Usage of Strengthening Rules

Strengthening useful for:

- Basic symmetry breaking
- Without-loss-of-generality reasoning
- Introducing extension variables

Additional strengthening rule:

- So-called dominance-based strengthening rule for advanced symmetry breaking
- See [BGMN23] for details

Usage of Strengthening Rules

Strengthening useful for:

- Basic symmetry breaking
- Without-loss-of-generality reasoning
- Introducing extension variables

Additional strengthening rule:

- So-called dominance-based strengthening rule for advanced symmetry breaking
- See [BGMN23] for details
- ... or next talk by Markus Anders about "Proof logging for symmetry breaking"

Deletion

Problem:

- Deleting constraints arbitrarily is unsound, as
  - Introduce better than optimal solutions
  - Even remove all solutions (when combined with dominance-based strengthening)
- Deletion needs to be restricted

Deletion

Problem:

- Deleting constraints arbitrarily is unsound, as
  - Introduce better than optimal solutions
  - Even remove all solutions (when combined with dominance-based strengthening)
- Deletion needs to be restricted

Solution:

- Constraint C can only be deleted if
  - C in derived set $\mathcal{D}$
  - C rederivable by redundance-based strengthening from core set $\mathcal{C}$ without using C

$$(\mathcal{C} \setminus \{C\}) \cup \{\neg C\} \vdash ((\mathcal{C} \setminus \{C\}) \cup C){\restriction_\omega} \cup \{f \geq f{\restriction_\omega}\}$$

Deletion: Example

$$\begin{aligned} \min \quad & x_1 + x_2 \\ \text{s.t.} \quad & x_1 + 2x_2 + 3\overline{x}_3 \geq 3 && (42) \\ & \overline{x}_3 \geq 1 && (43) \end{aligned} \quad \longrightarrow \quad \begin{aligned} \min \quad & x_1 + x_2 \\ \text{s.t.} \quad & \overline{x}_3 \geq 1 && (43) \end{aligned}$$

```
del id 42 ;
```

- Deletion of constraint `42` with empty substitution
- Deleted constraint implied by propagation

$$(\mathcal{C} \setminus \{C\}) \cup \{\neg C\} \vdash ((\mathcal{C} \setminus \{C\}) \cup C) \cup \{f \geq f\}$$

Deletion: Example

$$\begin{aligned} \min \quad & x_1 + x_2 \\ \text{s.t.} \quad & x_1 + 2x_2 + 3\bar{x}_3 \geq 3 \quad (42) \\ & \bar{x}_3 \geq 1 \quad (43) \end{aligned} \longrightarrow \begin{aligned} \min \quad & x_1 + x_2 \\ \text{s.t.} \quad & \bar{x}_3 \geq 1 \quad (43) \end{aligned}$$

```
del id 42 ;
```

- Deletion of constraint `42` with empty substitution
- Deleted constraint implied by propagation

$$(\mathcal{C} \setminus \{C\}) \cup \{\neg C\} \vdash ((\mathcal{C} \setminus \{C\}) \cup C) \cup \{f \geq f\}$$

- Also with explicit subproof (proof by contradiction)

`del id 42 : : subproof`	Add negated constraint $\bar{x}_1 + 2\bar{x}_2 + 3x_3 \geq 4$
`pol -1 43 3 * + ;`	Cutting planes proof resulting in $\bar{x}_1 + 2\bar{x}_2 \geq 4$
`qed : -1 ;`	Previous constraint is contradiction

Objective Update Rule

Effect:

- Allows objective function change from f_{old} to f_{new}
- Required by many reformulation techniques

Objective Update Rule

Effect:

- Allows objective function change from f_{old} to f_{new}
- Required by many reformulation techniques

Check:

- Equality $f_{old} = f_{new}$ trivial or explicit cutting planes proof
- Only core constraints can be used for this check
  - Deriving $f_{old} \leq f_{new}$ from the derived set can introduce better than optimal solutions

Objective Update Rule

Effect:

- Allows objective function change from f_{old} to f_{new}
- Required by many reformulation techniques

Check:

- Equality $f_{old} = f_{new}$ trivial or explicit cutting planes proof
- Only core constraints can be used for this check
  - Deriving $f_{old} \leq f_{new}$ from the derived set can introduce better than optimal solutions

Objective update specification options:

1. Specify new objective f_{new}
   - Good if change is large or new objective small
2. Specify difference between new and old objective $f_{new} - f_{old}$
   - Good for small objective changes

Objective Update Rule Necessary?

Without objective update:

- Redundance-based strengthening becomes more complicated to impossible

Objective Update Rule Necessary?

Without objective update:

- Redundance-based strengthening becomes more complicated to impossible

Example:

$$\begin{aligned} \min \quad & x_1 + x_2 \\ \text{s.t.} \quad & x_1 + x_2 + \bar{x}_3 + \bar{x}_4 \geq 2 \\ & x_1 + x_2 + \bar{x}_3 + \bar{x}_4 \leq 2 \\ & \bar{x}_3 + \bar{x}_4 \geq 1 \end{aligned}$$

Objective Update Rule Necessary?

Without objective update:

- Redundance-based strengthening becomes more complicated to impossible

Example:

$$\begin{aligned} \min \quad & x_1 + x_2 \\ \text{s.t.} \quad & x_1 + x_2 = x_3 + x_4 \\ & \overline{x}_3 + \overline{x}_4 \geq 1 \end{aligned}$$

Objective Update Rule Necessary?

Without objective update:

- Redundance-based strengthening becomes more complicated to impossible

Example:

$$\begin{aligned} \min \quad & x_1 + x_2 \\ \text{s.t.} \quad & x_1 + x_2 = x_3 + x_4 \\ & \bar{x}_3 + \bar{x}_4 \geq 1 \end{aligned} \longrightarrow \begin{aligned} \min \quad & x_3 + x_4 \\ \text{s.t.} \quad & x_1 + x_2 = x_3 + x_4 \\ & \bar{x}_3 + \bar{x}_4 \geq 1 \end{aligned}$$

- Deletion of $x_1 + x_2 + \bar{x}_3 + \bar{x}_4 \geq 2$ with substitution $\{x_1 \mapsto 1\}$

Objective Update Rule Necessary?

Without objective update:

- Redundance-based strengthening becomes more complicated to impossible

Example:

$$\begin{aligned} \min\quad & x_1 + x_2 \\ \text{s.t.}\quad & x_1 + x_2 = x_3 + x_4 \\ & \overline{x}_3 + \overline{x}_4 \geq 1 \end{aligned} \quad \longrightarrow \quad \begin{aligned} \min\quad & x_3 + x_4 \\ \text{s.t.}\quad & x_1 + x_2 = x_3 + x_4 \\ & \overline{x}_3 + \overline{x}_4 \geq 1 \end{aligned}$$

- Deletion of $x_1 + x_2 + \overline{x}_3 + \overline{x}_4 \geq 2$ with substitution $\{x_1 \mapsto 1\}$
- If objective unchanged, then $x_1 + x_2 \geq 1 + x_2$ ($f \geq f\restriction_\omega$) has to be shown
- This is not required if objective is updated, as $x_3 + x_4 \geq x_3 + x_4$ ($f \geq f\restriction_\omega$) is trivial

Objective Update: Example

$$\begin{aligned} \min \quad & 2x_1 + 3\overline{x}_2 & \\ \text{s.t.} \quad & 3\overline{x}_2 + 2\overline{x}_3 \geq 3 & (1) \\ & 3x_2 + 2x_3 \geq 2 & (2) \end{aligned} \longrightarrow \begin{aligned} \min \quad & 2x_1 + 2x_3 + 1 & \\ \text{s.t.} \quad & 3\overline{x}_2 + 2\overline{x}_3 \geq 3 & (1) \\ & 3x_2 + 2x_3 \geq 2 & (2) \end{aligned}$$

- Constraint (1) says $3\overline{x}_2 \geq 2x_3 + 1$
- Constraint (2) says $3\overline{x}_2 \leq 2x_3 + 1$

Updating to a new objective:

```
obju new 2 x1 2 x3 1 ;
```

- Change objective to $2x_1 + 2x_3 + 1$

```
obju diff -3 ~x2 2 x3 1 ;
```

- Change objective with difference $f_{new} - f_{old} = 2x_3 + 1 - 3\overline{x_2}$
- New objective is $f_{new} = f_{old} + 2x_3 + 1 - 3\overline{x_2}$

Example: Start of Proof

Proof starts with a header specifying the format version:

```
pseudo-Boolean proof version 3.0
```

Example: Start of Proof

Proof starts with a header specifying the format version:

```
pseudo-Boolean proof version 3.0
```

Input problem is loaded, e.g., from OPB file, and IDs assigned to initial constraints:

```
min: 1 x1 1 x2 ;
1 x1 1 x2 1 ~x3 1 ~x4 >= 3 ;
1 ~x1 1 ~x2 1 x3 1 x4 >= 1 ;
1 ~x1 1 x5 >= 1 ;
```

$$\begin{aligned} \min \quad & x_1 + x_2 \\ \text{s.t.} \quad & x_1 + x_2 + \overline{x}_3 + \overline{x}_4 \geq 3 && (1) \\ & \overline{x}_1 + \overline{x}_2 + x_3 + x_4 \geq 1 && (2) \\ & \overline{x}_1 + x_5 \geq 1 && (3) \end{aligned}$$

Example (1/4): Substitution

$$
\begin{aligned}
\min \quad & x_1 + x_2 \\
\text{s.t.} \quad & x_1 + x_2 + \overline{x}_3 + \overline{x}_4 \geq 3 && (1) \\
& \overline{x}_1 + \overline{x}_2 + x_3 + x_4 \geq 1 && (2) \\
& \overline{x}_1 + x_5 \geq 1 && (3)
\end{aligned}
\quad \longrightarrow \quad
\begin{aligned}
\min \quad & x_1 + x_2 \\
\text{s.t.} \quad & x_1 + x_2 + \overline{x}_3 + \overline{x}_4 \geq 3 && (1) \\
& \overline{x}_1 + \overline{x}_2 + x_3 + x_4 \geq 1 && (2) \\
& x_2 + \overline{x}_3 + \overline{x}_4 + x_5 \geq 3 && (4)
\end{aligned}
$$

- Constraints (1) and (2) say that $x_1 = \overline{x}_2 + x_3 + x_4$
- Substitute x_1 in constraint (3) using this equality

Example (1/4): Substitution

$$\begin{aligned} \min \quad & x_1 + x_2 \\ \text{s.t.} \quad & x_1 + x_2 + \overline{x}_3 + \overline{x}_4 \geq 3 \quad (1) \\ & \overline{x}_1 + \overline{x}_2 + x_3 + x_4 \geq 1 \quad (2) \\ & \overline{x}_1 + x_5 \geq 1 \quad (3) \end{aligned} \longrightarrow \begin{aligned} \min \quad & x_1 + x_2 \\ \text{s.t.} \quad & x_1 + x_2 + \overline{x}_3 + \overline{x}_4 \geq 3 \quad (1) \\ & \overline{x}_1 + \overline{x}_2 + x_3 + x_4 \geq 1 \quad (2) \\ & x_2 + \overline{x}_3 + \overline{x}_4 + x_5 \geq 3 \quad (4) \end{aligned}$$

- Constraints (1) and (2) say that $x_1 = \overline{x}_2 + x_3 + x_4$
- Substitute x_1 in constraint (3) using this equality

Certification:

`pol 1 3 + ;`	Add up constraints (1) and (3) to derive (4)
`core id 4 ;`	Move constraint (4) to core set

Example (1/4): Substitution

$$\begin{aligned} \min \quad & x_1 + x_2 \\ \text{s.t.} \quad & x_1 + x_2 + \bar{x}_3 + \bar{x}_4 \geq 3 && (1) \\ & \bar{x}_1 + \bar{x}_2 + x_3 + x_4 \geq 1 && (2) \\ & \bar{x}_1 + x_5 \geq 1 && (3) \end{aligned} \longrightarrow \begin{aligned} \min \quad & x_1 + x_2 \\ \text{s.t.} \quad & x_1 + x_2 + \bar{x}_3 + \bar{x}_4 \geq 3 && (1) \\ & \bar{x}_1 + \bar{x}_2 + x_3 + x_4 \geq 1 && (2) \\ & x_2 + \bar{x}_3 + \bar{x}_4 + x_5 \geq 3 && (4) \end{aligned}$$

- Constraints (1) and (2) say that $x_1 = \bar{x}_2 + x_3 + x_4$
- Substitute x_1 in constraint (3) using this equality

Certification:

`pol 1 3 + ;`	Add up constraints (1) and (3) to derive (4)
`core id 4 ;`	Move constraint (4) to core set
`del id 3 : : subproof`	Add negation of constraint (3) to get $x_1 + \bar{x}_5 \geq 2$
`  pol -1 2 + ;`	Add (2) to previous constraint to get $\bar{x}_2 + x_3 + x_4 + \bar{x}_5 \geq 2$
`  pol -1 4 + ;`	Add (4) to previous constraint to get $0 \geq 1$
`qed : -1 ;`	End subproof to delete constraint (3)

Example (2/4): Objective Function Update

$$\begin{aligned} \min \quad & x_1 + x_2 \\ \text{s.t.} \quad & x_1 + x_2 + \bar{x}_3 + \bar{x}_4 \geq 3 \quad (1) \\ & \bar{x}_1 + \bar{x}_2 + x_3 + x_4 \geq 1 \quad (2) \\ & x_2 + \bar{x}_3 + \bar{x}_4 + x_5 \geq 3 \quad (4) \end{aligned} \longrightarrow \begin{aligned} \min \quad & x_3 + x_4 + 1 \\ \text{s.t.} \quad & x_1 + x_2 + \bar{x}_3 + \bar{x}_4 \geq 3 \quad (1) \\ & \bar{x}_1 + \bar{x}_2 + x_3 + x_4 \geq 1 \quad (2) \\ & x_2 + \bar{x}_3 + \bar{x}_4 + x_5 \geq 3 \quad (4) \end{aligned}$$

- Change objective from $x_1 + x_2$ to $x_3 + x_4 + 1$ using constraints (1) and (2)
  - Constraint (1) says $x_1 + x_2 \geq x_3 + x_4 + 1$
  - Constraint (2) says $x_1 + x_2 \leq x_3 + x_4 + 1$

Example (2/4): Objective Function Update

$$\begin{aligned} \min\quad & x_1 + x_2 \\ \text{s.t.}\quad & x_1 + x_2 + \overline{x}_3 + \overline{x}_4 \geq 3 \quad (1) \\ & \overline{x}_1 + \overline{x}_2 + x_3 + x_4 \geq 1 \quad (2) \\ & x_2 + \overline{x}_3 + \overline{x}_4 + x_5 \geq 3 \quad (4) \end{aligned} \longrightarrow \begin{aligned} \min\quad & x_3 + x_4 + 1 \\ \text{s.t.}\quad & x_1 + x_2 + \overline{x}_3 + \overline{x}_4 \geq 3 \quad (1) \\ & \overline{x}_1 + \overline{x}_2 + x_3 + x_4 \geq 1 \quad (2) \\ & x_2 + \overline{x}_3 + \overline{x}_4 + x_5 \geq 3 \quad (4) \end{aligned}$$

- Change objective from $x_1 + x_2$ to $x_3 + x_4 + 1$ using constraints (1) and (2)
  - Constraint (1) says $x_1 + x_2 \geq x_3 + x_4 + 1$
  - Constraint (2) says $x_1 + x_2 \leq x_3 + x_4 + 1$

Certification:

`obju new 1 x3 1 x4 1 ;` | Change objective to $x_3 + x_4 + 1$ using (1) and (2)

Example (3/4): Delete Substitution Constraints (1/2)

$$\begin{aligned} \min \quad & x_3 + x_4 + 1 \\ \text{s.t.} \quad & x_1 + x_2 + \overline{x}_3 + \overline{x}_4 \geq 3 \quad (1) \\ & \overline{x}_1 + \overline{x}_2 + x_3 + x_4 \geq 1 \quad (2) \\ & x_2 + \overline{x}_3 + \overline{x}_4 + x_5 \geq 3 \quad (4) \end{aligned} \longrightarrow \begin{aligned} \min \quad & x_3 + x_4 + 1 \\ \text{s.t.} \quad & \overline{x}_1 + \overline{x}_2 + x_3 + x_4 \geq 1 \quad (2) \\ & x_2 + \overline{x}_3 + \overline{x}_4 + x_5 \geq 3 \quad (4) \end{aligned}$$

- Constraints (1) and (2) can be deleted, as they define $x_1 = \overline{x}_2 + x_3 + x_4$

Example (3/4): Delete Substitution Constraints (1/2)

$$\begin{aligned} \min \quad & x_3 + x_4 + 1 \\ \text{s.t.} \quad & x_1 + x_2 + \overline{x}_3 + \overline{x}_4 \geq 3 \quad (1) \\ & \overline{x}_1 + \overline{x}_2 + x_3 + x_4 \geq 1 \quad (2) \\ & x_2 + \overline{x}_3 + \overline{x}_4 + x_5 \geq 3 \quad (4) \end{aligned} \longrightarrow \begin{aligned} \min \quad & x_3 + x_4 + 1 \\ \text{s.t.} \quad & \overline{x}_1 + \overline{x}_2 + x_3 + x_4 \geq 1 \quad (2) \\ & x_2 + \overline{x}_3 + \overline{x}_4 + x_5 \geq 3 \quad (4) \end{aligned}$$

- Constraints (1) and (2) can be deleted, as they define $x_1 = \overline{x}_2 + x_3 + x_4$

Certification:

```
del id 1 : x1 -> 1 ;
```

Negation of (1) is $\overline{x}_1 + \overline{x}_2 + x_3 + x_4 \geq 2$, hence:
For (1), $x_2 + \overline{x}_3 + \overline{x}_4 \geq 2$ is implied by (4);
For (2), $\overline{x}_2 + x_3 + x_4 \geq 1$ is implied by negated constraint;
For (4), substitution does not change the constraint;
For $f \geq f\restriction_{\{x_1 \mapsto 1\}}$, is trivial, since x_1 not in objective

Example (3/4): Delete Substitution Constraints (2/2)

$$\begin{aligned} \min \quad & x_3 + x_4 + 1 \\ \text{s.t.} \quad & \bar{x}_1 + \bar{x}_2 + x_3 + x_4 \geq 1 \quad (2) \\ & x_2 + \bar{x}_3 + \bar{x}_4 + x_5 \geq 3 \quad (4) \end{aligned} \longrightarrow \begin{aligned} \min \quad & x_3 + x_4 + 1 \\ \text{s.t.} \quad & x_2 + \bar{x}_3 + \bar{x}_4 + x_5 \geq 3 \quad (4) \end{aligned}$$

- Now also delete constraint (2)

Certification:

```
del id 2 : x1 -> 0 ;
```

Delete (2) using substitution $\{x_1 \mapsto 0\}$:
For (2), $\bar{x}_2 + x_3 + x_4 \geq 0$ is trivial;
(4) and $f \geq f\restriction_{\{x_1 \mapsto 0\}}$ are again trivial

Example (4/4): Duality-Based Fixing

$$\begin{aligned} \min \quad & x_3 + x_4 + 1 \\ \text{s.t.} \quad & x_2 + \overline{x}_3 + \overline{x}_4 + x_5 \geq 3 \quad (4) \end{aligned} \longrightarrow \begin{aligned} \min \quad & x_3 + x_4 + 1 \\ \text{s.t.} \quad & \emptyset \end{aligned}$$

- W.l.o.g. x_2/x_5 can be fixed to 1, as
  - x_2/x_5 only appear positive in constraints with non-negative coefficient
  - objective coefficients for x_2/x_5 are 0
- W.l.o.g. x_3/x_4 can be fixed to 0, as
  - x_3/x_4 only appear negative in constraints with non-negative coefficient
  - x_3/x_4 only appear positive in the objective with non-negative coefficient

Example (4/4): Duality-Based Fixing

$$\begin{array}{ll} \min & x_3 + x_4 + 1 \\ \text{s.t.} & x_2 + \bar{x}_3 + \bar{x}_4 + x_5 \geq 3 \quad (4) \end{array} \longrightarrow \begin{array}{ll} \min & x_3 + x_4 + 1 \\ \text{s.t.} & \emptyset \end{array}$$

- W.l.o.g. x_2/x_5 can be fixed to 1, as
  - x_2/x_5 only appear positive in constraints with non-negative coefficient
  - objective coefficients for x_2/x_5 are 0
- W.l.o.g. x_3/x_4 can be fixed to 0, as
  - x_3/x_4 only appear negative in constraints with non-negative coefficient
  - x_3/x_4 only appear positive in the objective with non-negative coefficient

Certification:

```
del id 4 : x2 -> 1 x3 -> 0
  x4 -> 0 x5 -> 1 ;
```

Delete (4) using
$\omega = \{x_2 \mapsto 1, x_3 \mapsto 0, x_4 \mapsto 0, x_5 \mapsto 1\}$,
as $(x_2 + \bar{x}_3 + \bar{x}_4 + x_5 \geq 3)\restriction_\omega$ is $0 \geq -1$ and
$f \geq f\restriction_\omega$ is $x_3 + x_4 \geq -1$, which are both trivial

Example: Concluding Proof

We claim at the end of the proof that resulting formula has same optimal value:

```
output EQUIOPTIMAL FILE ;
```

- Check that core set and objective is syntactically equivalent to reformulated problem
- `FILE` means that reformulated problem is given in additional file

Example: Concluding Proof

We claim at the end of the proof that resulting formula has same optimal value:

```
output EQUIOPTIMAL FILE ;
```

- Check that core set and objective is syntactically equivalent to reformulated problem
- `FILE` means that reformulated problem is given in additional file

There is no conclusion regarding solving the instance:

```
conclusion NONE ;
```

Example: Concluding Proof

We claim at the end of the proof that resulting formula has same optimal value:

```
output EQUIOPTIMAL FILE ;
```

- Check that core set and objective is syntactically equivalent to reformulated problem
- `FILE` means that reformulated problem is given in additional file

There is no conclusion regarding solving the instance:

```
conclusion NONE ;
```

End the proof:

```
end pseudo-Boolean proof ;
```

Example: Full Proof

reformulation.proof

```
pseudo-Boolean proof version 3.0
pol 1 3 + ;
core id 4 ;
del id 3 : : subproof
   pol -1 2 + ;
   pol -1 4 + ;
qed : -1 ;
obju new 1 x3 1 x4 1 ;
del id 1 : x1 -> 1 ;
del id 2 : x1 -> 0 ;
del id 4 : x2 -> 1 x3 -> 0 x4 -> 0 x5 -> 1 ;
output EQUIOPTIMAL FILE ;
conclusion NONE ;
end pseudo-Boolean proof ;
```

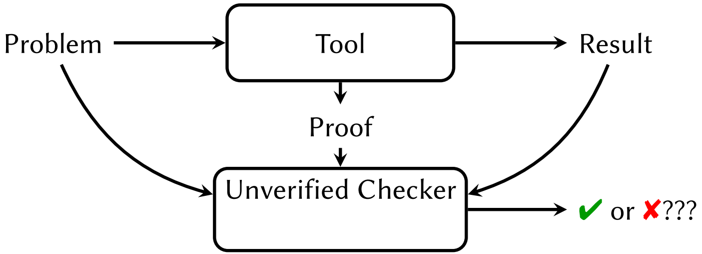
input.opb

```
min: 1 x1 1 x2 ;
1 x1 1 x2 1 ~x3 1 ~x4 >= 3 ;
1 ~x1 1 ~x2 1 x3 1 x4 >= 1 ;
1 ~x1 1 x5 >= 1 ;
```

output.opb

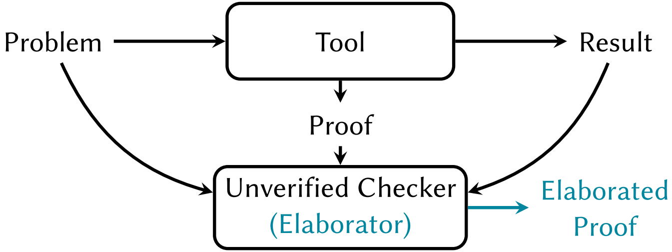
```
min: 1 x3 1 x4 1 ;
```

Formally Verified Proof Checking



How can we trust the checker?

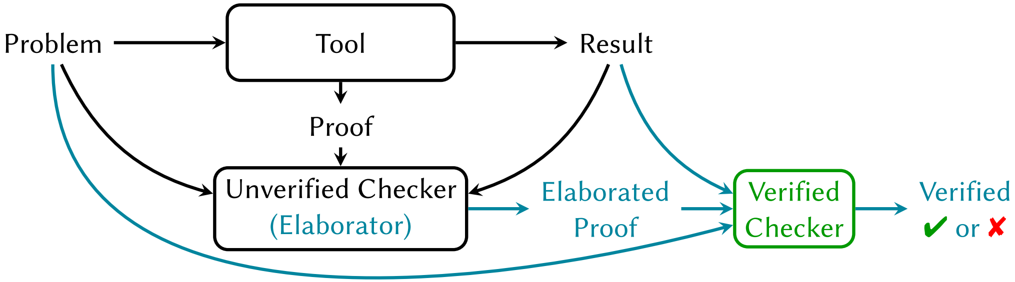
Formally Verified Proof Checking



How can we trust the checker?

1. Tool generates proof, which contains syntactic sugar for easy logging
2. Unverified proof checker elaborates syntactic sugar to simpler elaborated proof

Formally Verified Proof Checking



How can we trust the checker?

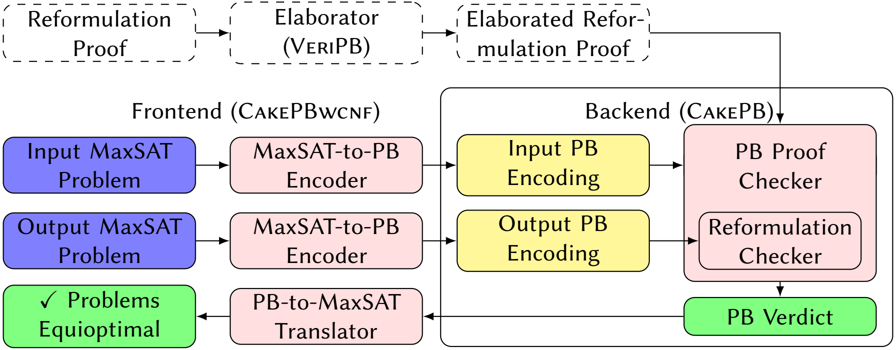
1. Tool generates proof, which contains syntactic sugar for easy logging
2. Unverified proof checker elaborates syntactic sugar to simpler elaborated proof
3. Elaborated proof checked by formally verified checker

Formal Verification Trust Base

What we have to trust:

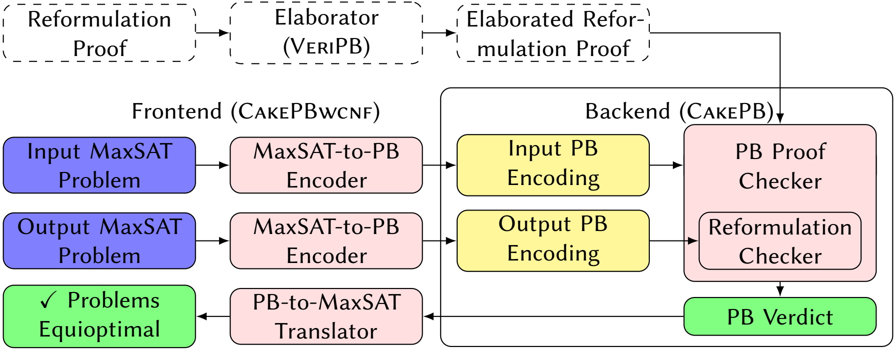
- Higher-order logic (HOL) definitions of parser and problems
  - kept as simple as possible, easy to check
- HOL model of CakeML environment and correspondence to real system
  - has been validated extensively
- HOL4 theorem prover, including its logic, implementation, and execution environment
  - separated and trustworthy kernel checks every logical inference

Framework for Formally Verified Reformulation Checkers



- Example workflow for checking MaxSAT preprocessing proofs
- But easy to adapt checker to new problem domain, e.g., 0-1 ILP, graph problems, ...

Framework for Formally Verified Reformulation Checkers



- Example workflow for checking MaxSAT preprocessing proofs
- But easy to adapt checker to new problem domain, e.g., 0-1 ILP, graph problems, ...
- Talk on CakePB tomorrow 9:00 by Yong Kiam Tan

Experimental Setup 0-1 ILP Presolving [HOGN24]

Tools:

- Added pseudo-Boolean proof logging to presolver PAPILO[1]
  - All presolve reductions applied to 0-1 ILPs in PAPILO covered
- Proof checked using proof checker VERIPB[2]

[1] https://github.com/scipopt/papilo

[2] https://gitlab.com/MIAOresearch/software/VeriPB

Experimental Setup 0-1 ILP Presolving [HOGN24]

Tools:

- Added pseudo-Boolean proof logging to presolver PaPILO[1]
    - All presolve reductions applied to 0-1 ILPs in PaPILO covered
- Proof checked using proof checker VeriPB[2]

Benchmarks:

- PB competition 2016 instances [Pse16]
- MIPLIB17 instances translated to OPB format [Dev20]

[1] https://github.com/scipopt/papilo
[2] https://gitlab.com/MIAOresearch/software/VeriPB

Proof Logging Overhead in PaPILO

Test set	size	default [s]	w/proof log [s]	relative
PB16-dec	1397	0.06	0.06	1.00
MIPLIB01-dec	291	0.42	0.43	1.02
PB16-opt	531	0.65	0.66	1.02
MIPLIB01-opt	142	0.33	0.35	1.06

- Additional time required to write proof is very small
- For 99% of instances less than 0.001s per reduction for certification

Certificate Checking Performance

			PaPILO time [s]		VeriPB	relative time w.r.t.	
test set	size	verified	default	w/proof log	time [s]	default	w/proof log
PB-dec	1397	1397	0.06	0.06	0.88	14.67	14.67
MIPLIB-dec	291	267	0.42	0.43	9.64	22.85	22.42
PB-opt	531	520	0.65	0.66	10.44	16.06	15.82
MIPLIB-opt	142	139	0.33	0.35	5.25	15.91	15.00

- Most instances verified within 10 000s timeout
- Overhead can be explained by PaPILO having more context than VeriPB
- PaPILO parallelizes some tasks, VeriPB works only sequentially
- Old version of VeriPB used, as new version[3] does not yet support required features

[3] https://gitlab.com/MIAOresearch/software/pboxide

Experimental Setup MaxSAT Preprocessing [IOT+24]

Tools:

- Added pseudo-Boolean proof logging to MaxSAT preprocessor MaxPre[4]
  - All techniques in MaxPre covered
- Proofs elaborated by VeriPB[2]
- Elaborated proofs checked by formally verified checker CakePB[5]

[4] https://bitbucket.org/coreo-group/maxpre2
[2] https://gitlab.com/MIAOresearch/software/VeriPB
[5] https://gitlab.com/MIAOresearch/software/cakepb

Experimental Setup MaxSAT Preprocessing [IOT+24]

Tools:

- Added pseudo-Boolean proof logging to MaxSAT preprocessor MaxPre[4]
  - All techniques in MaxPre covered
- Proofs elaborated by VeriPB[2]
- Elaborated proofs checked by formally verified checker CakePB[5]

Benchmarks:

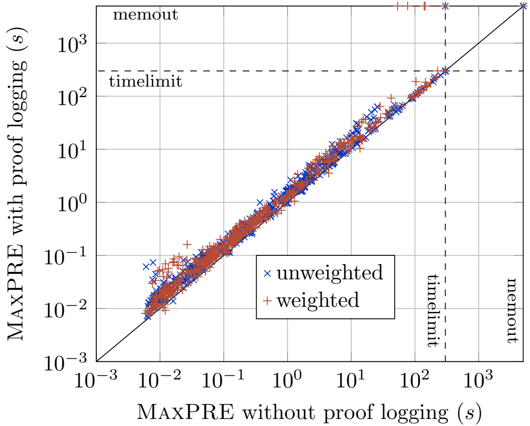
- MaxSAT evaluation 2023 instances [Max23]

---

[4] https://bitbucket.org/coreo-group/maxpre2
[2] https://gitlab.com/MIAOresearch/software/VeriPB
[5] https://gitlab.com/MIAOresearch/software/cakepb

Proof Logging Overhead in MaxPre

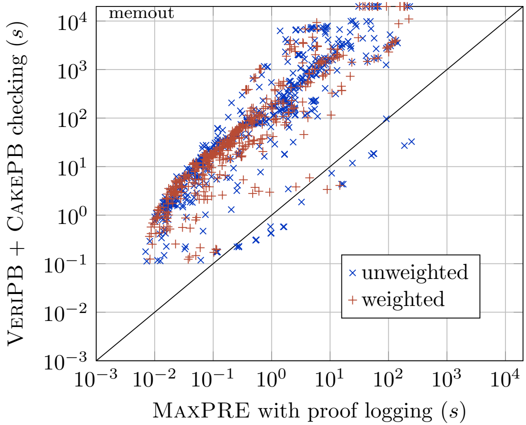
MaxPre with and without proof logging:



- 46% slower with proof logging
- Larger overhead than for 0-1 ILP presolving
- Bottleneck: Renaming of variables required for MaxSAT file format

Certificate Checking Performance

MaxPre vs. formally verified checking:



- 92% of instances checked
- Renaming of variables also bottleneck
- Elaboration with VeriPB 6.7× slower than CakePB
- Old version of VeriPB used
  - New VeriPB version should improve performance significantly
  - CakePB has been also been improved

Conclusion & Future Directions

Summary:

- Proof logging for reformulating optimization problems is possible with VeriPB
  - Can justify preprocessing/presolving techniques
  - Rules preserve optimal value
  - Deleting constraints requires care
  - Also special rule for updating objectives required
- Proof logging for standalone reformulation tools
- Formally verified end-to-end verification for problem reformulations

Conclusion & Future Directions

Summary:

- Proof logging for reformulating optimization problems is possible with VeriPB
  - Can justify preprocessing/presolving techniques
  - Rules preserve optimal value
  - Deleting constraints requires care
  - Also special rule for updating objectives required
- Proof logging for standalone reformulation tools
- Formally verified end-to-end verification for problem reformulations

Future research directions:

- Proof logging for MIP presolving (integer variables, rational coefficients) [DEGH23]
- Generalize reformulation proofs to enumeration and counting problems

Conclusion & Future Directions

Summary:

- Proof logging for reformulating optimization problems is possible with VeriPB
  - Can justify preprocessing/presolving techniques
  - Rules preserve optimal value
  - Deleting constraints requires care
  - Also special rule for updating objectives required
- Proof logging for standalone reformulation tools
- Formally verified end-to-end verification for problem reformulations

Future research directions:

- Proof logging for MIP presolving (integer variables, rational coefficients) [DEGH23]
- Generalize reformulation proofs to enumeration and counting problems

Thank you for your attention!

References I

[ABG+20] Tobias Achterberg, Robert E. Bixby, Zonghao Gu, Edward Rothberg, and Dieter Weninger.
Presolve reductions in mixed integer programming.
INFORMS Journal on Computing, 32(2):473–506, 2020.

[BBN+23] Jeremias Berg, Bart Bogaerts, Jakob Nordström, Andy Oertel, and Dieter Vandesande.
Certified core-guided MaxSAT solving.
In *Proceedings of the 29th International Conference on Automated Deduction (CADE-29)*, volume 14132 of *Lecture Notes in Computer Science*, pages 1–22. Springer, July 2023.

[BBN+24] Jeremias Berg, Bart Bogaerts, Jakob Nordström, Andy Oertel, Tobias Paxian, and Dieter Vandesande.
Certifying without loss of generality reasoning in solution-improving maximum satisfiability.
In *Proceedings of the 30th International Conference on Principles and Practice of Constraint Programming (CP '24)*, volume 307 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 4:1–4:28, September 2024.

[BGMN23] Bart Bogaerts, Stephan Gocht, Ciaran McCreesh, and Jakob Nordström.
Certified dominance and symmetry breaking for combinatorial optimisation.
Journal of Artificial Intelligence Research, 77:1539–1589, August 2023.
Preliminary version in *AAAI '22*.

References II

[BT19] Samuel R. Buss and Neil Thapen.
DRAT proofs, propagation redundancy, and extended resolution.
In *Proceedings of the 22nd International Conference on Theory and Applications of Satisfiability Testing (SAT '19)*, volume 11628 of *Lecture Notes in Computer Science*, pages 71–89. Springer, July 2019.

[CCT87] William Cook, Collette Rene Coullard, and György Turán.
On the complexity of cutting-plane proofs.
Discrete Applied Mathematics, 18(1):25–38, November 1987.

[CGS17] Kevin K. H. Cheung, Ambros M. Gleixner, and Daniel E. Steffy.
Verifying integer programming results.
In *Proceedings of the 19th International Conference on Integer Programming and Combinatorial Optimization (IPCO '17)*, volume 10328 of *Lecture Notes in Computer Science*, pages 148–160. Springer, June 2017.

[DEGH23] Jasper van Doornmalen, Leon Eifler, Ambros Gleixner, and Christopher Hojny.
A proof system for certifying symmetry and optimality reasoning in integer programming.
Technical Report 2311.03877, arXiv.org, November 2023.

[Dev20] Jo Devriendt.
Miplib 0-1 instances in opb format, May 2020.

References III

[GN21] Stephan Gocht and Jakob Nordström.
Certifying parity reasoning efficiently using pseudo-Boolean proofs.
In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI '21)*, pages 3768–3777, February 2021.

[HOGN24] Alexander Hoen, Andy Oertel, Ambros Gleixner, and Jakob Nordström.
Certifying MIP-based presolve reductions for 0–1 integer linear programs.
In *Proceedings of the 21st International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR '24)*, volume 14742 of *Lecture Notes in Computer Science*, pages 310–328. Springer, May 2024.

[IBJ22] Hannes Ihalainen, Jeremias Berg, and Matti Järvisalo.
Clause redundancy and preprocessing in maximum satisfiability.
In *Proceedings of the 11th International Joint Conference on Automated Reasoning (IJCAR '22)*, volume 13385 of *Lecture Notes in Computer Science*, pages 75–94. Springer, August 2022.

[IOT+24] Hannes Ihalainen, Andy Oertel, Yong Kiam Tan, Jeremias Berg, Matti Järvisalo, Magnus O. Myreen, and Jakob Nordström.
Certified MaxSAT preprocessing.
In *Proceedings of the 12th International Joint Conference on Automated Reasoning (IJCAR '24)*, volume 14739 of *Lecture Notes in Computer Science*, pages 396–418. Springer, July 2024.

References IV

[Max23] MaxSAT evaluation 2023.
`https://maxsat-evaluations.github.io/2023/`, July 2023.

[Pse16] Pseudo-Boolean competition 2016.
`https://www.cril.univ-artois.fr/PB16/`, July 2016.

[Van23] Dieter Vandesande.
Towards certified MaxSAT solving: Certified MaxSAT solving with SAT oracles and encodings of pseudo-Boolean constraints.
Master's thesis, Vrije Universiteit Brussel (VUB), 2023.

[VDB22] Dieter Vandesande, Wolf De Wulf, and Bart Bogaerts.
QMaxSATpb: A certified MaxSAT solver.
In *Proceedings of the 16th International Conference on Logic Programming and Non-monotonic Reasoning (LPNMR '22)*, volume 13416 of *Lecture Notes in Computer Science*, pages 429–442. Springer, September 2022.