Practically Feasible Proof Logging for Pseudo-Boolean Optimization

Wietze Koops

Lund University and University of Copenhagen

NordConsNet workshop, Uppsala, Sweden
August 21, 2025

Joint work with Daniel Le Berre, Magnus O. Myreen, Jakob Nordström, Andy Oertel, Yong Kiam Tan, and Marc Vinyals

Published in CP '25

SAT and Combinatorial Solving

- Impressive progress in SAT solving over last couple of decades [BHvMW21]
- Also big successes in more expressive paradigms:
  - Constraint programming
  - Satisfiability modulo theories (SMT)
  - Mixed integer linear programming (MILP)

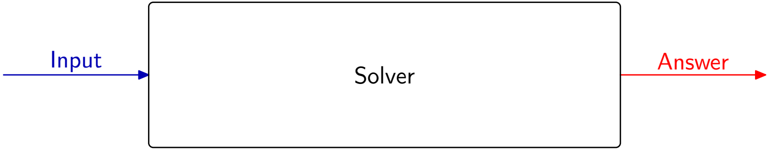
SAT and Combinatorial Solving

- Impressive progress in SAT solving over last couple of decades [BHvMW21]
- Also big successes in more expressive paradigms:
  - Constraint programming
  - Satisfiability modulo theories (SMT)
  - Mixed integer linear programming (MILP)
- However, solvers are sometimes wrong (even best commercial ones) [CGS17, AGJ+18, GSD19, GCS23, BBN+23, Tin24]

SAT and Combinatorial Solving

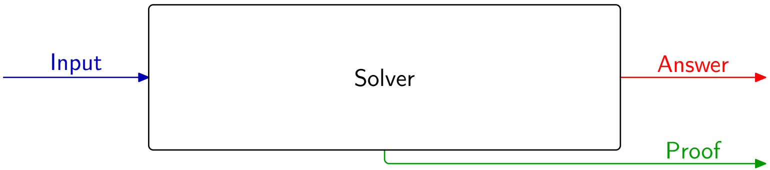
- Impressive progress in SAT solving over last couple of decades [BHvMW21]
- Also big successes in more expressive paradigms:
  - Constraint programming
  - Satisfiability modulo theories (SMT)
  - Mixed integer linear programming (MILP)
- However, solvers are sometimes wrong (even best commercial ones) [CGS17, AGJ+18, GSD19, GCS23, BBN+23, Tin24]
- Most successful solution: certifying algorithms with proof logging

Proof Logging with Certifying Solvers: Workflow



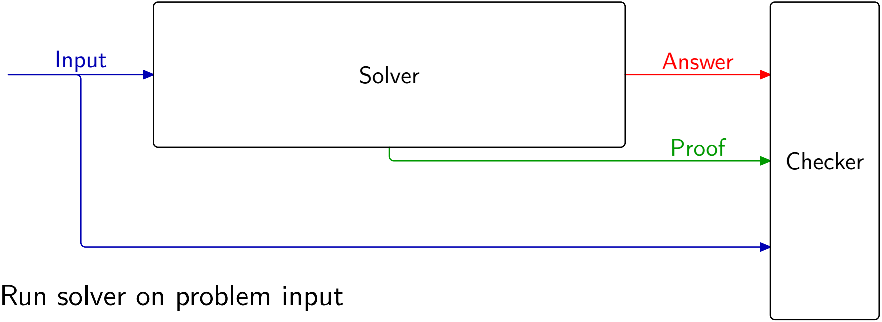
1. Run solver on problem input

Proof Logging with Certifying Solvers: Workflow



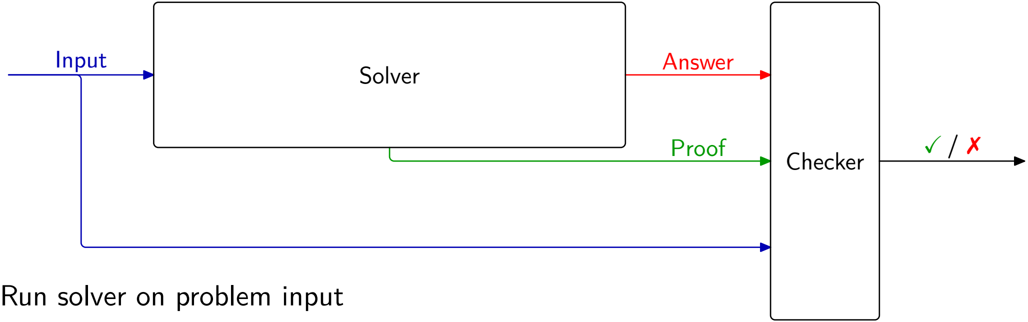
1. Run solver on problem input
2. Get as output not only answer but also proof

Proof Logging with Certifying Solvers: Workflow



1. Run solver on problem input
2. Get as output not only answer but also proof
3. Feed input + answer + proof to proof checker

Proof Logging with Certifying Solvers: Workflow



1. Run solver on problem input
2. Get as output not only answer but also proof
3. Feed input + answer + proof to proof checker
4. Verify that proof checker says answer is correct

Proof Logging for SAT Solving and Beyond

- Proof logging for SAT: big success story [HHW13, WHH14, CHH+17, BCH21]
- Great performance:
  - Small constant overhead for proof generation ($\lessapprox 10\%$ of solving time)
  - Efficient proof checking ($\lessapprox 10\times$ solving time)

Proof Logging for SAT Solving and Beyond

- Proof logging for SAT: big success story [HHW13, WHH14, CHH+17, BCH21]
- Great performance:
  - Small constant overhead for proof generation ($\lesssim 10\%$ of solving time)
  - Efficient proof checking ($\lesssim 10\times$ solving time)
- More expressive paradigms: VeriPB proof logging
  - MaxSAT solving [VDB22, BBN+23, IOT+24, BBN+24]
  - Subgraph solving [GMN20, GMM+20, GMM+24]
  - Constraint programming [EGMN20, GMN22, MM23, MMN24, MM25]
  - Automated planning [DHN+25]
  - and more ...

Proof Logging for SAT Solving and Beyond

- Proof logging for SAT: big success story [HHW13, WHH14, CHH+17, BCH21]
- Great performance:
  - Small constant overhead for proof generation ($\lesssim 10\%$ of solving time)
  - Efficient proof checking ($\lesssim 10\times$ solving time)
- More expressive paradigms: VeriPB proof logging
  - MaxSAT solving [VDB22, BBN+23, IOT+24, BBN+24]
  - Subgraph solving [GMN20, GMM+20, GMM+24]
  - Constraint programming [EGMN20, GMN22, MM23, MMN24, MM25]
  - Automated planning [DHN+25]
  - and more ...
- But much worse performance:
  - Proof logging overhead: sometimes $\times 10$ or worse
  - Proof checking overhead: sometimes $\times 1000$ or worse

Our Contribution

- Efficient VeriPB proof logging and checking for pseudo-Boolean optimization
- Covers all techniques in state-of-the-art solvers RoundingSat and Sat4j
- Including formally verified proof checking backend

Our Contribution

- Efficient VeriPB proof logging and checking for pseudo-Boolean optimization
- Covers all techniques in state-of-the-art solvers RoundingSat and Sat4j
- Including formally verified proof checking backend
- Performance close to expectations for SAT solving:
  - Proof logging overhead usually $\leqslant 10\%$ (worst-case 50%)
  - Checking overhead usually $\leqslant \times 6$ (worst-case $\times 20$)
- First time practically feasible logging for combinatorial optimization beyond SAT

Pseudo-Boolean Optimization

- Operates on 0-1 integer linear inequalities or pseudo-Boolean constraints:

$$\sum_i a_i \ell_i \geq A$$

  - $a_i, A \in \mathbb{Z}$
  - literals ℓ_i: x_i or $\overline{x}_i$ (where $x_i + \overline{x}_i = 1$)
  - variables x_i take values $0 = \mathit{false}$ or $1 = \mathit{true}$

- Objective $\sum_i w_i \ell_i$ to be minimized (for maximization, negate objective)

Pseudo-Boolean Optimization

- Operates on 0-1 integer linear inequalities or pseudo-Boolean constraints:

$$\sum_i a_i \ell_i \geq A$$

  - $a_i, A \in \mathbb{Z}$
  - literals ℓ_i: x_i or $\overline{x}_i$ (where $x_i + \overline{x}_i = 1$)
  - variables x_i take values $0 = \mathit{false}$ or $1 = \mathit{true}$

- Objective $\sum_i w_i \ell_i$ to be minimized (for maximization, negate objective)

- Examples of pseudo-Boolean constraints:
  - Clauses: $x_1 \vee x_2 \vee \overline{x_3} \quad \Longleftrightarrow \quad x_1 + x_2 + \overline{x_3} \geq 1$
  - Cardinality constraints: $x_1 + x_2 + x_3 \geq 2$
  - General constraints: $3x_1 + 2x_2 + x_3 + x_4 \geq 3$

Approaches for Pseudo-Boolean Optimization

- Two main approaches:
  - Translate to CNF and run conflict-driven clause learning (CDCL)
  - Generalize conflict-driven search to pseudo-Boolean inequalities (our focus)

Approaches for Pseudo-Boolean Optimization

- Two main approaches:
  - Translate to CNF and run conflict-driven clause learning (CDCL)
  - Generalize conflict-driven search to pseudo-Boolean inequalities (our focus)
- New challenges and techniques compared to SAT:
  - Efficient propagation [Dev20, NORZ24]
  - Linear programming (LP) integration [DGN21]
  - Optimization techniques, e.g. solution-improving search, core-guided search [DGD+21]

Proof Logging for Pseudo-Boolean Optimization

- Conflict analysis:
  - In SAT, learned clauses are checked using reverse unit propagation (RUP)
  - In PB, explicit reasoning steps are needed

Proof Logging for Pseudo-Boolean Optimization

- Conflict analysis:
  - In SAT, learned clauses are checked using reverse unit propagation (RUP)
  - In PB, explicit reasoning steps are needed
- Other techniques pose further challenges:
  - Objective rewriting in core-guided search
  - Linear programming (LP) integration (Farkas certificates, cut generation, ...)

Proof Logging for Pseudo-Boolean Optimization

- Conflict analysis:
  - In SAT, learned clauses are checked using reverse unit propagation (RUP)
  - In PB, explicit reasoning steps are needed
- Other techniques pose further challenges:
  - Objective rewriting in core-guided search
  - Linear programming (LP) integration (Farkas certificates, cut generation, ...)
- Challenges for efficient proof logging:
  - Logging unit constraints (saying that a variable must take some fixed value)
  - Logging constraint simplifications (e.g. simplifying away fixed values)
  - Logging and checking solutions
  - Formally verified proof checking

Pseudo-Boolean Proof Logging Basics

Pseudo-Boolean proof logging based on cutting planes proof system [CCT87]

Input axioms From the input

Pseudo-Boolean Proof Logging Basics

Pseudo-Boolean proof logging based on cutting planes proof system [CCT87]

Input axioms From the input

Literal axioms $\overline{\ell_i \geq 0}$

Pseudo-Boolean Proof Logging Basics

Pseudo-Boolean proof logging based on cutting planes proof system [CCT87]

Input axioms — From the input

Literal axioms

$$\frac{}{\ell_i \geq 0}$$

Addition

$$\frac{\sum_i a_i \ell_i \geq A \qquad \sum_i b_i \ell_i \geq B}{\sum_i (a_i + b_i)\ell_i \geq A + B}$$

Pseudo-Boolean Proof Logging Basics

Pseudo-Boolean proof logging based on cutting planes proof system [CCT87]

Input axioms — From the input

Literal axioms

$$\frac{}{\ell_i \geq 0}$$

Addition

$$\frac{\sum_i a_i \ell_i \geq A \qquad \sum_i b_i \ell_i \geq B}{\sum_i (a_i + b_i)\ell_i \geq A + B}$$

Multiplication for any $c \in \mathbb{N}^+$

$$\frac{\sum_i a_i \ell_i \geq A}{\sum_i c a_i \ell_i \geq cA}$$

Pseudo-Boolean Proof Logging Basics

Pseudo-Boolean proof logging based on cutting planes proof system [CCT87]

Input axioms From the input

Literal axioms

$$\frac{}{\ell_i \geq 0}$$

Addition

$$\frac{\sum_i a_i \ell_i \geq A \qquad \sum_i b_i \ell_i \geq B}{\sum_i (a_i + b_i)\ell_i \geq A + B}$$

Multiplication for any $c \in \mathbb{N}^+$

$$\frac{\sum_i a_i \ell_i \geq A}{\sum_i c a_i \ell_i \geq cA}$$

Division for any $c \in \mathbb{N}^+$

$$\frac{\sum_i a_i \ell_i \geq A}{\sum_i \left\lceil \frac{a_i}{c} \right\rceil \ell_i \geq \left\lceil \frac{A}{c} \right\rceil}$$

Cutting Planes Toy Example

$$w + 2x + y \geq 2$$

Cutting Planes Toy Example

Multiply by 2 $$\frac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4}$$

Cutting Planes Toy Example

Multiply by 2 $\dfrac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4}$ $\quad w + 2x + 4y + 2z \geq 5$

Cutting Planes Toy Example

$$\text{Multiply by } 2 \; \dfrac{\dfrac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4} \qquad w + 2x + 4y + 2z \geq 5}{3w + 6x + 6y + 2z \geq 9} \; \text{Add}$$

Cutting Planes Toy Example

$$\text{Multiply by } 2 \ \frac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4}$$

$$\text{Add} \ \frac{2w + 4x + 2y \geq 4 \qquad w + 2x + 4y + 2z \geq 5}{3w + 6x + 6y + 2z \geq 9} \qquad \overline{z} \geq 0$$

Cutting Planes Toy Example

Multiply by 2
$$\frac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4}$$

Add
$$\frac{2w + 4x + 2y \geq 4 \qquad w + 2x + 4y + 2z \geq 5}{3w + 6x + 6y + 2z \geq 9}$$

$$\frac{\overline{z} \geq 0}{2\overline{z} \geq 0}$$
Multiply by 2

Cutting Planes Toy Example

$$
\dfrac{\dfrac{\dfrac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4}\ \text{Multiply by } 2 \qquad w + 2x + 4y + 2z \geq 5}{3w + 6x + 6y + 2z \geq 9}\ \text{Add} \qquad \dfrac{\overline{z} \geq 0}{2\overline{z} \geq 0}\ \text{Multiply by } 2}{3w + 6x + 6y + 2z + 2\overline{z} \geq 9}\ \text{Add}
$$

Cutting Planes Toy Example

$$\dfrac{\dfrac{\dfrac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4}\ \text{Multiply by } 2 \qquad w + 2x + 4y + 2z \geq 5}{3w + 6x + 6y + 2z \geq 9}\ \text{Add} \qquad \dfrac{\overline{z} \geq 0}{2\overline{z} \geq 0}\ \text{Multiply by } 2}{3w + 6x + 6y + 2 \qquad \geq 9}\ \text{Add}$$

Cutting Planes Toy Example

$$
\text{Add}\ \dfrac{\text{Multiply by 2}\ \dfrac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4} \qquad w + 2x + 4y + 2z \geq 5}{3w + 6x + 6y + 2z \geq 9}
$$

$$
\text{Add}\ \dfrac{3w + 6x + 6y + 2z \geq 9 \qquad \dfrac{\overline{z} \geq 0}{2\overline{z} \geq 0}\ \text{Multiply by 2}}{3w + 6x + 6y \qquad \geq 7}
$$

Cutting Planes Toy Example

$$
\text{Divide by } 3\;\dfrac{\text{Add}\;\dfrac{\text{Add}\;\dfrac{\text{Multiply by } 2\;\dfrac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4} \qquad w + 2x + 4y + 2z \geq 5}{3w + 6x + 6y + 2z \geq 9} \qquad \dfrac{\overline{z} \geq 0}{2\overline{z} \geq 0}\;\text{Multiply by } 2}{3w + 6x + 6y \geq 7}}{w + 2x + 2y \geq 2\tfrac{1}{3}}
$$

Cutting Planes Toy Example

Multiply by 2: $\dfrac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4}$

Add: $\dfrac{2w + 4x + 2y \geq 4 \qquad w + 2x + 4y + 2z \geq 5}{3w + 6x + 6y + 2z \geq 9}$

Multiply by 2: $\dfrac{\overline{z} \geq 0}{2\overline{z} \geq 0}$

Add: $\dfrac{3w + 6x + 6y + 2z \geq 9 \qquad 2\overline{z} \geq 0}{3w + 6x + 6y \geq 7}$

Divide by 3: $\dfrac{3w + 6x + 6y \geq 7}{w + 2x + 2y \geq 3}$

Cutting Planes Toy Example

$$
\dfrac{
  \dfrac{
    \dfrac{\dfrac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4}\ \text{Multiply by } 2 \qquad w + 2x + 4y + 2z \geq 5}{3w + 6x + 6y + 2z \geq 9}\ \text{Add}
    \qquad
    \dfrac{\overline{z} \geq 0}{2\overline{z} \geq 0}\ \text{Multiply by } 2
  }{3w + 6x + 6y \geq 7}\ \text{Add}
}{w + 2x + 2y \geq 3}\ \text{Divide by } 3
$$

By naming constraints by integers and literal axioms by the literal involved as

$$
\begin{aligned}
\text{Constraint @C1} &\doteq w + 2x + y \geq 2 \\
\text{Constraint @C2} &\doteq w + 2x + 4y + 2z \geq 5 \\
\sim\mathtt{z} &\doteq \overline{z} \geq 0
\end{aligned}
$$

Cutting Planes Toy Example

$$\text{Multiply by } 2 \quad \frac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4}$$

$$\text{Add} \quad \frac{2w + 4x + 2y \geq 4 \qquad w + 2x + 4y + 2z \geq 5}{3w + 6x + 6y + 2z \geq 9}$$

$$\frac{\overline{z} \geq 0}{2\overline{z} \geq 0} \quad \text{Multiply by } 2$$

$$\text{Add} \quad \frac{3w + 6x + 6y + 2z \geq 9 \qquad 2\overline{z} \geq 0}{3w + 6x + 6y \geq 7}$$

$$\text{Divide by } 3 \quad \frac{3w + 6x + 6y \geq 7}{w + 2x + 2y \geq 3}$$

By naming constraints by integers and literal axioms by the literal involved as

$$\text{Constraint @C1} \doteq w + 2x + y \geq 2$$

$$\text{Constraint @C2} \doteq w + 2x + 4y + 2z \geq 5$$

$$\sim\!\texttt{z} \doteq \overline{z} \geq 0$$

such a calculation is written in the proof log in reverse Polish notation as

```
pol @C1 2 * @C2 + ~z 2 * + 3 d
```

Advanced Pseudo-Boolean Proof Logging

We need a rule for deriving non-implied constraints (e.g. introducing new variables)

Redundance-based strengthening ([BT19, GN21], inspired by [JHB12], simplified)

F and $F \cup \{C\}$ are equisatisfiable if there is a substitution ω (mapping variables to truth values or literals), called a witness, for which

$$F \cup \{\neg C\} \models (F \cup \{C\})\restriction_\omega$$

Advanced Pseudo-Boolean Proof Logging

We need a rule for deriving non-implied constraints (e.g. introducing new variables)

Redundance-based strengthening ([BT19, GN21], inspired by [JHB12], simplified)

F and $F \cup \{C\}$ are equisatisfiable if there is a substitution ω (mapping variables to truth values or literals), called a witness, for which

$$F \cup \{\neg C\} \models (F \cup \{C\})\restriction_{\omega}$$

- Proof sketch: If α satisfies F but falsifies C, then $\alpha \circ \omega$ satisfies $F \cup \{C\}$

Advanced Pseudo-Boolean Proof Logging

We need a rule for deriving non-implied constraints (e.g. introducing new variables)

Redundance-based strengthening ([BT19, GN21], inspired by [JHB12], simplified)

F and $F \cup \{C\}$ are equisatisfiable if there is a substitution ω (mapping variables to truth values or literals), called a witness, for which

$$F \cup \{\neg C\} \models (F \cup \{C\})\restriction_\omega$$

- Proof sketch: If α satisfies F but falsifies C, then $\alpha \circ \omega$ satisfies $F \cup \{C\}$
- In a proof, the implication needs to be efficiently verifiable — every $D \in (F \cup \{C\})\restriction_\omega$ should follow from $F \cup \{\neg C\}$ either
  1. "obviously" or
  2. by explicitly presented derivation

Redundance Rule: Example from Logging Core-Guided Search

Suppose we know $D \doteq x_1 + x_2 + x_3 \geq 2$.

Redundance Rule: Example from Logging Core-Guided Search

Suppose we know $D \doteq x_1 + x_2 + x_3 \geq 2$.
Want to introduce variable y_3 such that

$$x_1 + x_2 + x_3 = 2 + y_3, \quad \text{i.e.} \quad \begin{cases} C_1 & \doteq \; x_1 + x_2 + x_3 \leq 2 + y_3 \\ C_2 & \doteq \; x_1 + x_2 + x_3 \geq 2 + y_3 \end{cases}$$

using condition $F \cup \{\neg C\} \models (F \cup \{C\})\restriction_\omega$.

Redundance Rule: Example from Logging Core-Guided Search

Suppose we know $D \doteq x_1 + x_2 + x_3 \geq 2$.
Want to introduce variable y_3 such that

$$x_1 + x_2 + x_3 = 2 + y_3, \quad \text{i.e.} \quad \begin{cases} C_1 & \doteq \quad x_1 + x_2 + x_3 \leq 2 + y_3 \\ C_2 & \doteq \quad x_1 + x_2 + x_3 \geq 2 + y_3 \end{cases}$$

using condition $F \cup \{\neg C\} \models (F \cup \{C\})\restriction_\omega$.

- $F \cup \{\neg C_1\} \models (F \cup \{C_1\})\restriction_\omega$

Redundance Rule: Example from Logging Core-Guided Search

Suppose we know $D \doteq x_1 + x_2 + x_3 \geq 2$.
Want to introduce variable y_3 such that

$$x_1 + x_2 + x_3 = 2 + y_3, \quad \text{i.e.} \quad \begin{cases} C_1 & \doteq \; x_1 + x_2 + x_3 \leq 2 + y_3 \\ C_2 & \doteq \; x_1 + x_2 + x_3 \geq 2 + y_3 \end{cases}$$

using condition $F \cup \{\neg C\} \models (F \cup \{C\})\restriction_\omega$.

- $F \cup \{\neg C_1\} \models (F \cup \{C_1\})\restriction_\omega$
  Choose $\omega = \{y_3 \mapsto 1\}$ — F untouched; new constraint C_1 trivially satisfied

Redundance Rule: Example from Logging Core-Guided Search

Suppose we know $D \doteq x_1 + x_2 + x_3 \geq 2$.
Want to introduce variable y_3 such that

$$x_1 + x_2 + x_3 = 2 + y_3, \quad \text{i.e.} \quad \begin{cases} C_1 & \doteq \quad x_1 + x_2 + x_3 \leq 2 + y_3 \\ C_2 & \doteq \quad x_1 + x_2 + x_3 \geq 2 + y_3 \end{cases}$$

using condition $F \cup \{\neg C\} \models (F \cup \{C\})\restriction_\omega$.

- $F \cup \{\neg C_1\} \models (F \cup \{C_1\})\restriction_\omega$
  Choose $\omega = \{y_3 \mapsto 1\}$ — F untouched; new constraint C_1 trivially satisfied
- $F \cup \{C_1\} \cup \{\neg C_2\} \models (F \cup \{C_1\} \cup \{C_2\})\restriction_\omega$

Redundance Rule: Example from Logging Core-Guided Search

Suppose we know $D \doteq x_1 + x_2 + x_3 \geq 2$.
Want to introduce variable y_3 such that

$$x_1 + x_2 + x_3 = 2 + y_3, \quad \text{i.e.} \quad \begin{cases} C_1 \doteq x_1 + x_2 + x_3 \leq 2 + y_3 \\ C_2 \doteq x_1 + x_2 + x_3 \geq 2 + y_3 \end{cases}$$

using condition $F \cup \{\neg C\} \models (F \cup \{C\})\restriction_\omega$.

- $F \cup \{\neg C_1\} \models (F \cup \{C_1\})\restriction_\omega$
  Choose $\omega = \{y_3 \mapsto 1\}$ — F untouched; new constraint C_1 trivially satisfied
- $F \cup \{C_1\} \cup \{\neg C_2\} \models (F \cup \{C_1\} \cup \{C_2\})\restriction_\omega$
  Choose $\omega = \{y_3 \mapsto 0\}$ — F untouched; new constraint C_2 follows from D
  $\neg C_2 \doteq x_1 + x_2 + x_3 \leq 1 + y_3$ implies $C_1\restriction_\omega \doteq x_1 + x_2 + x_3 \leq 2$

Redundance Rule: Example from Logging Core-Guided Search

Suppose we know $D \doteq x_1 + x_2 + x_3 \geq 2$.
Want to introduce variable y_3 such that

$$x_1 + x_2 + x_3 = 2 + y_3, \quad \text{i.e.} \quad \begin{cases} C_1 \doteq x_1 + x_2 + x_3 \leq 2 + y_3 \\ C_2 \doteq x_1 + x_2 + x_3 \geq 2 + y_3 \end{cases}$$

using condition $F \cup \{\neg C\} \models (F \cup \{C\})\restriction_\omega$.

- $F \cup \{\neg C_1\} \models (F \cup \{C_1\})\restriction_\omega$
  Choose $\omega = \{y_3 \mapsto 1\}$ — F untouched; new constraint C_1 trivially satisfied
- $F \cup \{C_1\} \cup \{\neg C_2\} \models (F \cup \{C_1\} \cup \{C_2\})\restriction_\omega$
  Choose $\omega = \{y_3 \mapsto 0\}$ — F untouched; new constraint C_2 follows from D
  $\neg C_2 \doteq x_1 + x_2 + x_3 \leq 1 + y_3$ implies $C_1\restriction_\omega \doteq x_1 + x_2 + x_3 \leq 2$

VeriPB:

```
red +1 x1 +1 x2 +1 x3 -1 y3 <= 2; y3 -> 1
red +1 x1 +1 x2 +1 x3 -1 y3 >= 2; y3 -> 0
```

Farkas Certificates

The constraints

$$\begin{aligned} \textcolor{red}{C_1} &\doteq \quad x_1 + \ \ x_2 + x_3 &&\geq 2 \\ \textcolor{red}{C_2} &\doteq \quad 3x_1 + 2x_2 + x_3 + x_4 &&\geq 3 \\ \textcolor{red}{C_3} &\doteq -2x_1 - 2x_2 - x_3 &&\geq -1 \end{aligned}$$

are unsatisfiable even over the reals.

Farkas Certificates

The constraints

$$
\begin{aligned}
C_1 &\doteq \quad x_1 + \; x_2 + x_3 \qquad \geq 2 \\
C_2 &\doteq \quad 3x_1 + 2x_2 + x_3 + x_4 \geq 3 \\
C_3 &\doteq -2x_1 - 2x_2 - x_3 \qquad \geq -1
\end{aligned}
$$

are unsatisfiable even over the reals.

Farkas certificate: positive linear combination of constraints (and literal axioms, e.g. $\overline{x}_4 \geq 0 \doteq -x_4 \geq -1$) proving this:

$$C_1 + C_2 + 2C_3 + (\overline{x_4} \geq 0) + (x_2 \geq 0) \doteq 0 \geq 2$$

is a contradiction.

Farkas Certificates

The constraints

$$\begin{aligned} C_1 &\doteq \quad\ x_1 + \ \ x_2 + x_3 \qquad \geq 2 \\ C_2 &\doteq \quad 3x_1 + 2x_2 + x_3 + x_4 \geq 3 \\ C_3 &\doteq -2x_1 - 2x_2 - x_3 \qquad \geq -1 \end{aligned}$$

are unsatisfiable even over the reals.

Farkas certificate: positive linear combination of constraints (and literal axioms, e.g. $\overline{x}_4 \geq 0 \ \doteq\ -x_4 \geq -1$) proving this:

$$C_1 + C_2 + 2C_3 + (\overline{x_4} \geq 0) + (x_2 \geq 0) \ \doteq\ 0 \geq 2$$

is a contradiction.

VeriPB: `pol @C1 @C2 + @C3 2 * + ~x4 + x2 +`

Cut Generation: Basics

- Cut generation:
  - Technique borrowed from MIP
  - Add constraint (*cut*) implied by input formula
  - Cuts away rational solution found by LP solver

Cut Generation: Basics

- Cut generation:
  - Technique borrowed from MIP
  - Add constraint (*cut*) implied by input formula
  - Cuts away rational solution found by LP solver
- Example: Minimize $x_1 + x_2 + x_3$ subject to

$$\begin{aligned} C_1 &\doteq x_1 + x_2 \geq 1 \\ C_2 &\doteq x_1 + x_3 \geq 1 \\ C_3 &\doteq x_2 + x_3 \geq 1 \end{aligned}$$

  - Rational optimum $x_1 = x_2 = x_3 = \frac{1}{2}$

Cut Generation: Basics

- Cut generation:
  - Technique borrowed from MIP
  - Add constraint (*cut*) implied by input formula
  - Cuts away rational solution found by LP solver
- Example: Minimize $x_1 + x_2 + x_3$ subject to

$$
\begin{aligned}
C_1 &\doteq x_1 + x_2 \geq 1 \\
C_2 &\doteq x_1 + x_3 \geq 1 \\
C_3 &\doteq x_2 + x_3 \geq 1
\end{aligned}
$$

  - Rational optimum $x_1 = x_2 = x_3 = \frac{1}{2}$
  - Adding C_1, C_2 and C_3 yields $2x_1 + 2x_2 + 2x_3 \geq 3$

Cut Generation: Basics

- Cut generation:
  - Technique borrowed from MIP
  - Add constraint (*cut*) implied by input formula
  - Cuts away rational solution found by LP solver
- Example: Minimize $x_1 + x_2 + x_3$ subject to

$$\begin{aligned} C_1 &\doteq x_1 + x_2 \geq 1 \\ C_2 &\doteq x_1 + x_3 \geq 1 \\ C_3 &\doteq x_2 + x_3 \geq 1 \end{aligned}$$

  - Rational optimum $x_1 = x_2 = x_3 = \frac{1}{2}$
  - Adding C_1, C_2 and C_3 yields $2x_1 + 2x_2 + 2x_3 \geq 3$
  - Cutting planes division by 2 yields $x_1 + x_2 + x_3 \geq 2$

Cut Generation: Basics

- Cut generation:
  - Technique borrowed from MIP
  - Add constraint (*cut*) implied by input formula
  - Cuts away rational solution found by LP solver
- Example: Minimize $x_1 + x_2 + x_3$ subject to

$$C_1 \doteq x_1 + x_2 \geq 1$$
$$C_2 \doteq x_1 + x_3 \geq 1$$
$$C_3 \doteq x_2 + x_3 \geq 1$$

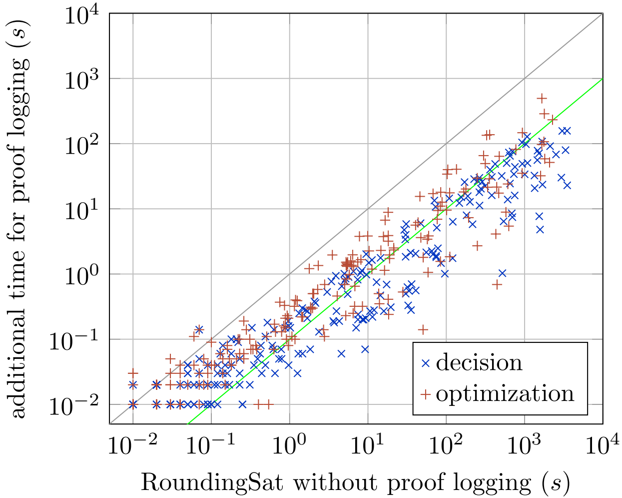
  - Rational optimum $x_1 = x_2 = x_3 = \frac{1}{2}$
  - Adding C_1, C_2 and C_3 yields $2x_1 + 2x_2 + 2x_3 \geq 3$
  - Cutting planes division by 2 yields $x_1 + x_2 + x_3 \geq 2$
  - VeriPB: `pol @C1 @C2 + @C3 + 2 d`

Advanced Cut Generation

- Cut generation with mixed integer rounding (MIR) rule [MW01, DGN21] more challenging
- Reasoning uses integer slack variables (not supported by VeriPB)
- Proof logging instead uses proof by contradiction

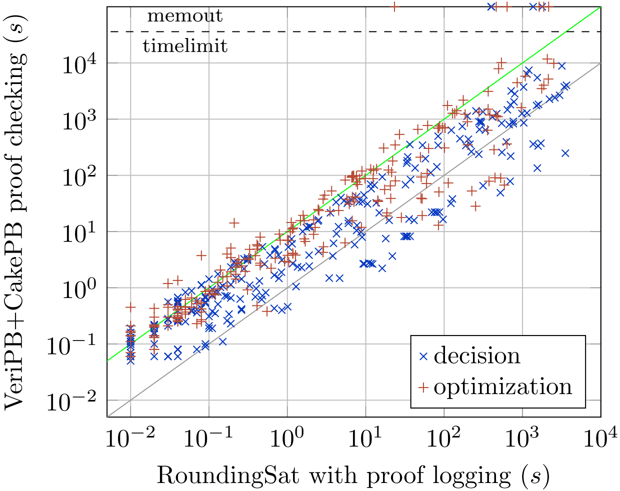
Empirical Results: Proof Logging Overhead

- Usually $\leqslant 10\%$
- Decision instances: worst-case 20%
- Optimization instances: worst-case 50%
- Overheads gets smaller for larger solving times



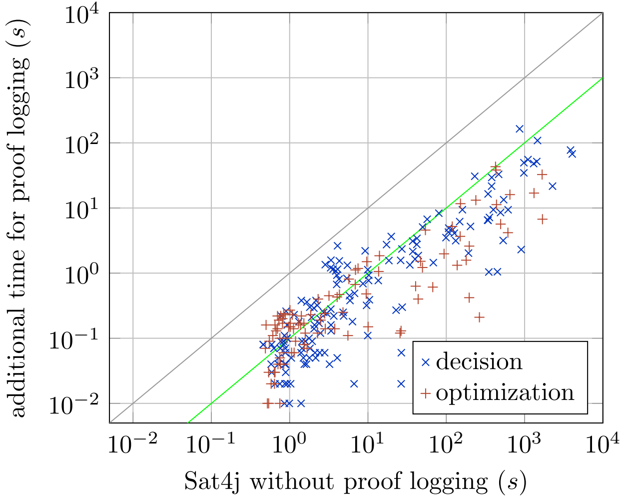
Empirical Results: Proof Checking Overhead

- Usually $\leqslant \times 6$
- Decision instances: worst-case $\times 10$
- Optimization instances: worst-case $\times 20$



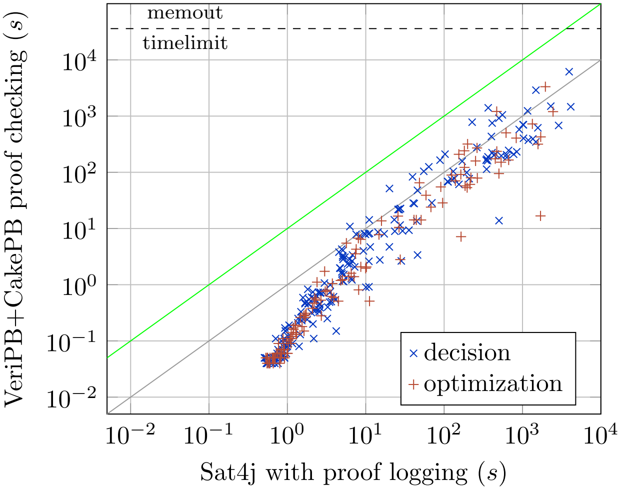
Empirical Results: Proof Logging Overhead Sat4j

- Usually $\leqslant 10\%$
- Worst-case 60%



Empirical Results: Proof Checking Overhead Sat4j

- Usually $\leqslant \times 2$
- Worst-case $\times 4$



Future Work

- Even faster proof logging and checking for pseudo-Boolean optimization
  - Branch-and-bound search (checking solutions currently a bottleneck)
  - Native efficient support for simplifications of constraints
  - Low-level optimizations in VeriPB and formally verified backend CakePB

Future Work

- Even faster proof logging and checking for pseudo-Boolean optimization
  - Branch-and-bound search (checking solutions currently a bottleneck)
  - Native efficient support for simplifications of constraints
  - Low-level optimizations in VeriPB and formally verified backend CakePB
- Faster proof logging and checking for further paradigms:
  - MaxSAT solving
  - Subgraph solving
  - Constraint programming
  - ...

Conclusion

- Efficient proof logging for pseudo-Boolean optimization using VeriPB
- First example of practically feasible certified solving beyond SAT
- Future directions:
  - Further improvements for pseudo-Boolean optimization
  - Efficient certified solving in other paradigms
- Is this the start of a new era: practically feasible proof logging beyond SAT?

Conclusion

- Efficient proof logging for pseudo-Boolean optimization using VeriPB
- First example of practically feasible certified solving beyond SAT
- Future directions:
  - Further improvements for pseudo-Boolean optimization
  - Efficient certified solving in other paradigms
- Is this the start of a new era: practically feasible proof logging beyond SAT?
- Thank you! Any questions?

References I

[AGJ+18] Özgür Akgün, Ian P. Gent, Christopher Jefferson, Ian Miguel, and Peter Nightingale. Metamorphic testing of constraint solvers. In *Proceedings of the 24th International Conference on Principles and Practice of Constraint Programming (CP '18)*, volume 11008 of *Lecture Notes in Computer Science*, pages 727–736. Springer, August 2018.

[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.

[BCH21] Seulkee Baek, Mario Carneiro, and Marijn J. H. Heule. A flexible proof format for SAT solver-elaborator communication. In *Proceedings of the 27th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS '21)*, volume 12651 of *Lecture Notes in Computer Science*, pages 59–75. Springer, March-April 2021.

References II

[BHvMW21] Armin Biere, Marijn J. H. Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2nd edition, February 2021.

[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.

[CHH+17] Luís Cruz-Filipe, Marijn J. H. Heule, Warren A. Hunt Jr., Matt Kaufmann, and Peter Schneider-Kamp. Efficient certified RAT verification. In *Proceedings of the 26th International Conference on Automated Deduction (CADE-26)*, volume 10395 of *Lecture Notes in Computer Science*, pages 220–236. Springer, August 2017.

References III

[Dev20] Jo Devriendt. Watched propagation of 0-1 integer linear constraints. In *Proceedings of the 26th International Conference on Principles and Practice of Constraint Programming (CP '20)*, volume 12333 of *Lecture Notes in Computer Science*, pages 160–176. Springer, September 2020.

[DGD+21] Jo Devriendt, Stephan Gocht, Emir Demirović, Jakob Nordström, and Peter Stuckey. Cutting to the core of pseudo-Boolean optimization: Combining core-guided search with cutting planes reasoning. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI '21)*, pages 3750–3758, February 2021.

[DGN21] Jo Devriendt, Ambros Gleixner, and Jakob Nordström. Learn to relax: Integrating 0-1 integer linear programming with pseudo-Boolean conflict-driven search. *Constraints*, 26(1–4):26–55, October 2021. Preliminary version in *CPAIOR '20*.

[DHN+25] Simon Dold, Malte Helmert, Jakob Nordström, Gabriele Röger, and Tanja Schindler. Pseudo-boolean proof logging for optimal classical planning. To appear in Proceedings of the 35th International Conference on Automated Planning and Scheduling (ICAPS '25), 2025.

[EGMN20] Jan Elffers, Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. Justifying all differences using pseudo-Boolean reasoning. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI '20)*, pages 1486–1494, February 2020.

References IV

[GCS23] Graeme Gange, Geoffrey Chu, and Peter J. Stuckey. Certifying optimality in constraint programming. Manuscript. Available at `https://people.eng.unimelb.edu.au/pstuckey/papers/certified-cp.pdf`, 2023.

[GMM+20] Stephan Gocht, Ross McBride, Ciaran McCreesh, Jakob Nordström, Patrick Prosser, and James Trimble. Certifying solvers for clique and maximum common (connected) subgraph problems. In *Proceedings of the 26th International Conference on Principles and Practice of Constraint Programming (CP '20)*, volume 12333 of *Lecture Notes in Computer Science*, pages 338–357. Springer, September 2020.

[GMM+24] Stephan Gocht, Ciaran McCreesh, Magnus O. Myreen, Jakob Nordström, Andy Oertel, and Yong Kiam Tan. End-to-end verification for subgraph solving. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI '24)*, pages 8038–8047, February 2024.

[GMN20] Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. Subgraph isomorphism meets cutting planes: Solving with certified solutions. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI '20)*, pages 1134–1140, July 2020.

References V

[GMN22] Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. An auditable constraint programming solver. In *Proceedings of the 28th International Conference on Principles and Practice of Constraint Programming (CP '22)*, volume 235 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:18, August 2022.

[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.

[GSD19] Xavier Gillard, Pierre Schaus, and Yves Deville. SolverCheck: Declarative testing of constraints. In *Proceedings of the 25th International Conference on Principles and Practice of Constraint Programming (CP '19)*, volume 11802 of *Lecture Notes in Computer Science*, pages 565–582. Springer, October 2019.

[HHW13] Marijn J. H. Heule, Warren A. Hunt Jr., and Nathan Wetzler. Trimming while checking clausal proofs. In *Proceedings of the 13th International Conference on Formal Methods in Computer-Aided Design (FMCAD '13)*, pages 181–188, October 2013.

References VI

[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.

[JHB12] Matti Järvisalo, Marijn J. H. Heule, and Armin Biere. Inprocessing rules. In *Proceedings of the 6th International Joint Conference on Automated Reasoning (IJCAR '12)*, volume 7364 of *Lecture Notes in Computer Science*, pages 355–370. Springer, June 2012.

[MM23] Matthew McIlree and Ciaran McCreesh. Proof logging for smart extensional constraints. In *Proceedings of the 29th International Conference on Principles and Practice of Constraint Programming (CP '23)*, volume 280 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:17, August 2023.

[MM25] Matthew McIlree and Ciaran McCreesh. Certifying bounds propagation for integer multiplication constraints. In *Proceedings of the 39th AAAI Conference on Artificial Intelligence (AAAI '25)*, pages 11309–11317, February-March 2025.

References VII

[MMN24] Matthew McIlree, Ciaran McCreesh, and Jakob Nordström. Proof logging for the circuit constraint. In *Proceedings of the 21st International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research (CPAIOR '24)*, volume 14743 of *Lecture Notes in Computer Science*, pages 38–55. Springer, May 2024.

[MW01] Hugues Marchand and Laurence A. Wolsey. Aggregation and mixed integer rounding to solve MIPs. *Operations Research*, 49(3):325–468, June 2001.

[NORZ24] Robert Nieuwenhuis, Albert Oliveras, Enric Rodríguez-Carbonell, and Rui Zhao. Speeding up pseudo-Boolean propagation. In *Proceedings of the 27th International Conference on Theory and Applications of Satisfiability Testing (SAT '24)*, volume 305 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:18, August 2024.

[Tin24] Cesare Tinelli. Scalable proof production and checking in SMT. In *Proceedings of the 27th International Conference on Theory and Applications of Satisfiability Testing (SAT '24)*, volume 305 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:2, August 2024.

References VIII

[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.

[WHH14] Nathan Wetzler, Marijn J. H. Heule, and Warren A. Hunt Jr. DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In *Proceedings of the 17th International Conference on Theory and Applications of Satisfiability Testing (SAT '14)*, volume 8561 of *Lecture Notes in Computer Science*, pages 422–429. Springer, July 2014.