

Proof Logging for Pseudo-Boolean Optimization

Wietze Koops

Lund University and University of Copenhagen

WHOOPS '25, Orsay, France
September 13, 2025

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



Proof Logging

- Proof logging is the most successful solution to deal with bugs in combinatorial solvers
- Big success story in Boolean satisfiability (SAT) [HHW13, WHH14, CHH⁺17, BCH21]

Proof Logging

- Proof logging is the most successful solution to deal with bugs in combinatorial solvers
- Big success story in Boolean satisfiability (SAT) [HHW13, WHH14, CHH⁺17, BCH21]
- Why is proof logging successful?

Proof Logging

- Proof logging is the most successful solution to deal with bugs in combinatorial solvers
- Big success story in Boolean satisfiability (SAT) [HHW13, WHH14, CHH⁺17, BCH21]
- Why is proof logging successful?
 - ▶ Expressivity: Wide or ideally full coverage of techniques in state-of-the-art solvers
Only need to add some simple print statements (and some bookkeeping)

Proof Logging

- Proof logging is the most successful solution to deal with bugs in combinatorial solvers
- Big success story in Boolean satisfiability (SAT) [HHW13, WHH14, CHH⁺17, BCH21]
- Why is proof logging successful?
 - ▶ Expressivity: Wide or ideally full coverage of techniques in state-of-the-art solvers
Only need to add some simple print statements (and some bookkeeping)
 - ▶ Efficiency: Fast proof logging and checking. SAT community standard:
 - ★ Small constant overhead for proof generation ($\lesssim 10\%$ of solving time)
 - ★ Efficient proof checking ($\lesssim 10\times$ solving time)

Proof Logging

- Proof logging is the most successful solution to deal with bugs in combinatorial solvers
- Big success story in Boolean satisfiability (SAT) [HHW13, WHH14, CHH⁺17, BCH21]
- Why is proof logging successful?
 - ▶ Expressivity: Wide or ideally full coverage of techniques in state-of-the-art solvers
Only need to add some simple print statements (and some bookkeeping)
 - ▶ Efficiency: Fast proof logging and checking. SAT community standard:
 - ★ Small constant overhead for proof generation ($\lesssim 10\%$ of solving time)
 - ★ Efficient proof checking ($\lesssim 10\times$ solving time)
 - ▶ Simplicity: Keep the proof system simple \implies Easy to write proof checker.

Proof Logging

- Proof logging is the most successful solution to deal with bugs in combinatorial solvers
- Big success story in Boolean satisfiability (SAT) [HHW13, WHH14, CHH⁺17, BCH21]
- Why is proof logging successful?
 - ▶ Expressivity: Wide or ideally full coverage of techniques in state-of-the-art solvers
Only need to add some simple print statements (and some bookkeeping)
 - ▶ Efficiency: Fast proof logging and checking. SAT community standard:
 - ★ Small constant overhead for proof generation ($\lesssim 10\%$ of solving time)
 - ★ Efficient proof checking ($\lesssim 10\times$ solving time)
 - ▶ Simplicity: Keep the proof system simple \implies Easy to write proof checker.
 - ▶ Trustworthiness:
 - ★ Proofs are fully complete, so each step easy to check
 - ★ Checker has a formally verified backend

Proof Logging: Existing Work Beyond SAT

- Constraint Programming:
 - ▶ Early work [VS10]: no full coverage, not trustworthy
 - ▶ *VeriPB* Proof Logging [EGMN20, GMN22, MM23, MMN24, MM25]: efficiency is a problem
(e.g. logging overhead $\times 10$, checking overhead $\times 1000$)

Proof Logging: Existing Work Beyond SAT

- Constraint Programming:
 - ▶ Early work [VS10]: no full coverage, not trustworthy
 - ▶ *VeriPB* Proof Logging [EGMN20, GMN22, MM23, MMN24, MM25]: efficiency is a problem (e.g. logging overhead $\times 10$, checking overhead $\times 1000$)
- Subgraph solving: *VeriPB* Proof Logging [GMM⁺20, GMM⁺24]: efficiency is a problem

Proof Logging: Existing Work Beyond SAT

- Constraint Programming:
 - ▶ Early work [VS10]: no full coverage, not trustworthy
 - ▶ *VeriPB* Proof Logging [EGMN20, GMN22, MM23, MMN24, MM25]: efficiency is a problem (e.g. logging overhead $\times 10$, checking overhead $\times 1000$)
- Subgraph solving: *VeriPB* Proof Logging [GMM⁺20, GMM⁺24]: efficiency is a problem
- Mixed-Integer Programming:
 - ▶ *VIPR* Proof Logging [CGS17]: not nearly full coverage
 - ▶ *VeriPB* Proof Logging (0–1 ILP Presolving) [HOGN24]: efficiency is a problem

Proof Logging: Existing Work Beyond SAT

- Constraint Programming:
 - ▶ Early work [VS10]: no full coverage, not trustworthy
 - ▶ *VeriPB* Proof Logging [EGMN20, GMN22, MM23, MMN24, MM25]: efficiency is a problem (e.g. logging overhead $\times 10$, checking overhead $\times 1000$)
- Subgraph solving: *VeriPB* Proof Logging [GMM⁺20, GMM⁺24]: efficiency is a problem
- Mixed-Integer Programming:
 - ▶ *VIPR* Proof Logging [CGS17]: not nearly full coverage
 - ▶ *VeriPB* Proof Logging (0–1 ILP Presolving) [HOGN24]: efficiency is a problem
- MaxSAT solving: *VeriPB* Proof Logging [BBN⁺23, BBN⁺24, IOT⁺24]: efficiency problem

Proof Logging: Existing Work Beyond SAT

- Constraint Programming:
 - ▶ Early work [VS10]: no full coverage, not trustworthy
 - ▶ *VeriPB* Proof Logging [EGMN20, GMN22, MM23, MMN24, MM25]: efficiency is a problem (e.g. logging overhead $\times 10$, checking overhead $\times 1000$)
- Subgraph solving: *VeriPB* Proof Logging [GMM⁺20, GMM⁺24]: efficiency is a problem
- Mixed-Integer Programming:
 - ▶ *VIPR* Proof Logging [CGS17]: not nearly full coverage
 - ▶ *VeriPB* Proof Logging (0–1 ILP Presolving) [HOGN24]: efficiency is a problem
- MaxSAT solving: *VeriPB* Proof Logging [BBN⁺23, BBN⁺24, IOT⁺24]: efficiency problem
- Model Counting: *KCPS* [Cap19], *CPOG* [BNAH23], *MICE* [FHR22]: efficiency problem

Proof Logging: Existing Work Beyond SAT

- Constraint Programming:
 - ▶ Early work [VS10]: no full coverage, not trustworthy
 - ▶ *VeriPB* Proof Logging [EGMN20, GMN22, MM23, MMN24, MM25]: efficiency is a problem (e.g. logging overhead $\times 10$, checking overhead $\times 1000$)
- Subgraph solving: *VeriPB* Proof Logging [GMM⁺20, GMM⁺24]: efficiency is a problem
- Mixed-Integer Programming:
 - ▶ *VIPR* Proof Logging [CGS17]: not nearly full coverage
 - ▶ *VeriPB* Proof Logging (0–1 ILP Presolving) [HOGN24]: efficiency is a problem
- MaxSAT solving: *VeriPB* Proof Logging [BBN⁺23, BBN⁺24, IOT⁺24]: efficiency problem
- Model Counting: *KCPS* [Cap19], *CPOG* [BNAH23], *MICE* [FHR22]: efficiency problem
- SMT solving: *Alethe* [SFBF21], *Carcara* [ALB23]: no full coverage, efficiency problem, and proof system is very complicated (100s of proof rules)

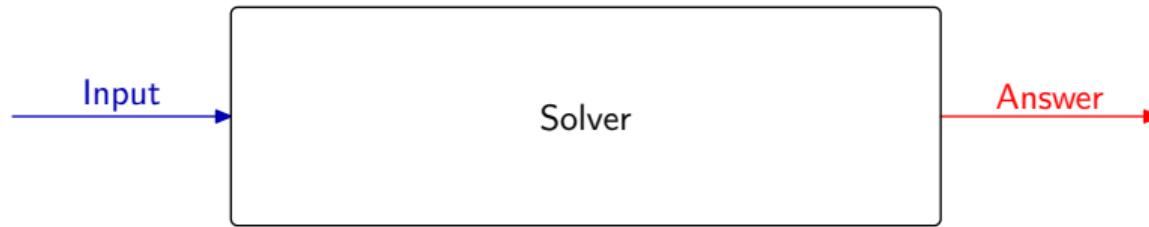
This Talk

- Efficient *VeriPB* proof logging and checking for pseudo-Boolean optimization [KLM⁺25]
- Covers all techniques in state-of-the-art solvers *RoundingSat* [EN18] and *Sat4j* [LP10]

This Talk

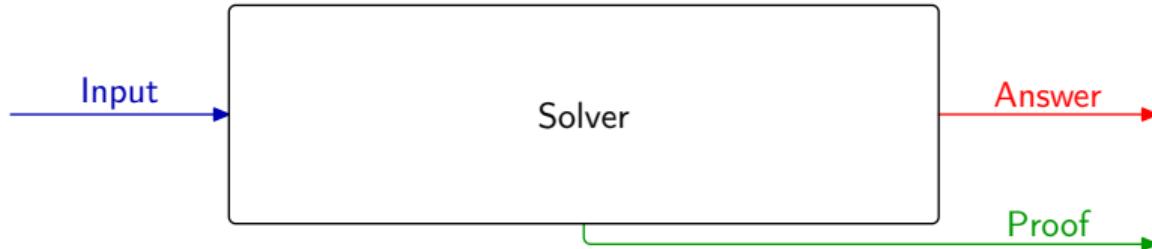
- Efficient *VeriPB* proof logging and checking for pseudo-Boolean optimization [KLM⁺25]
- Covers all techniques in state-of-the-art solvers *RoundingSat* [EN18] and *Sat4j* [LP10]
- Performance close to our goals:
 - ▶ Proof logging overhead usually $\leq 10\%$ (worst-case 50%)
 - ▶ Checking overhead usually $\leq \times 6$ (worst-case $\times 20$)
- First time practically feasible proof logging beyond SAT

Proof Logging with Certifying Solvers: Workflow



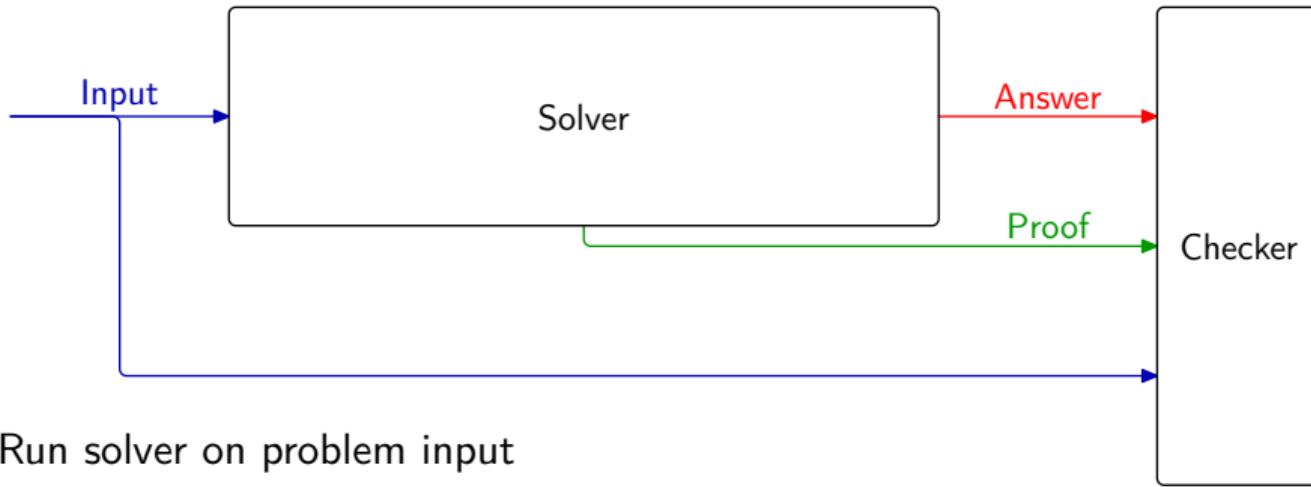
- ① Run solver on problem input

Proof Logging with Certifying Solvers: Workflow



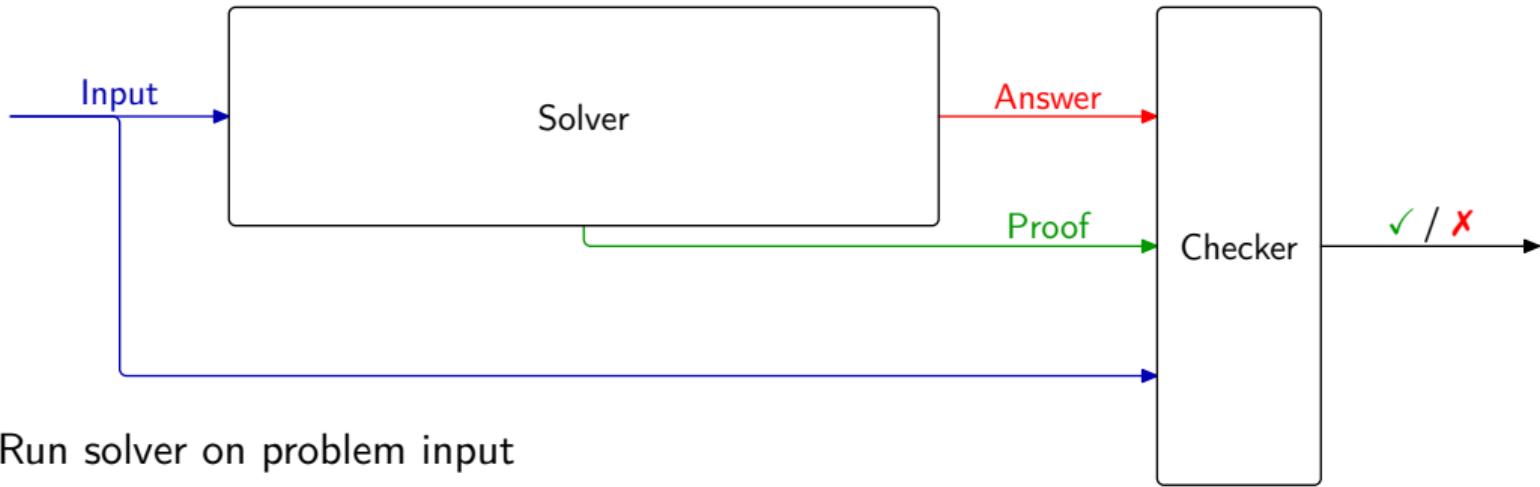
- ① Run solver on problem input
- ② Get as output not only answer but also proof

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

Overview of This Talk

1 Preliminaries

- Pseudo-Boolean Solving and Optimization
- The *VeriPB* Proof System
- Optimization Techniques

2 Proof Logging for Pseudo-Boolean Solving and Optimization

- Core-Guided Optimization
- LP Integration

3 Conclusion

- Empirical Results
- Take-Away Message

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 \bar{x}_i (where $x_i + \bar{x}_i = 1$)
- ▶ variables x_i take values 0 (false) or 1 (true)

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 \bar{x}_i (where $x_i + \bar{x}_i = 1$)
 - ▶ variables x_i take values 0 (false) or 1 (true)
-
- Objective $Obj = \sum_i w_i \ell_i$ 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 \bar{x}_i (where $x_i + \bar{x}_i = 1$)
 - ▶ variables x_i take values 0 (false) or 1 (true)
-
- Objective $Obj = \sum_i w_i \ell_i$ to be minimized (for maximization, negate objective)
 - Examples of pseudo-Boolean constraints:
 - ▶ Clauses: $x_1 \vee x_2 \vee \bar{x}_3 \iff x_1 + x_2 + \bar{x}_3 \geq 1$

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 \bar{x}_i (where $x_i + \bar{x}_i = 1$)
 - ▶ variables x_i take values 0 (false) or 1 (true)
- Objective $Obj = \sum_i w_i \ell_i$ to be minimized (for maximization, negate objective)
 - Examples of pseudo-Boolean constraints:
 - ▶ Clauses: $x_1 \vee x_2 \vee \bar{x}_3 \iff x_1 + x_2 + \bar{x}_3 \geq 1$
 - ▶ Cardinality constraints: $x_1 + x_2 + x_3 \geq 2$

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 \bar{x}_i (where $x_i + \bar{x}_i = 1$)
 - ▶ variables x_i take values 0 (false) or 1 (true)
-
- Objective $Obj = \sum_i w_i \ell_i$ to be minimized (for maximization, negate objective)
 - Examples of pseudo-Boolean constraints:
 - ▶ Clauses: $x_1 \vee x_2 \vee \bar{x}_3 \iff x_1 + x_2 + \bar{x}_3 \geq 1$
 - ▶ Cardinality constraints: $x_1 + x_2 + x_3 \geq 2$
 - ▶ General constraints: $3x_1 + 2x_2 + x_3 + x_4 \geq 3$

Conflict-Driven Search

- SAT and pseudo-Boolean solving: based on conflict-driven search

Conflict-Driven Search

- SAT and pseudo-Boolean solving: based on conflict-driven search
- Propagation: Infer literal values from single constraint and other literal values

Conflict-Driven Search

- SAT and pseudo-Boolean solving: based on conflict-driven search
- Propagation: Infer literal values from single constraint and other literal values
Example: After deciding $x_1 = 0$, constraint $3x_1 + 2x_2 + x_3 + x_4 \geq 3$ propagates $x_2 = 1$

Conflict-Driven Search

- SAT and pseudo-Boolean solving: based on conflict-driven search
- Propagation: Infer literal values from single constraint and other literal values
Example: After deciding $x_1 = 0$, constraint $3x_1 + 2x_2 + x_3 + x_4 \geq 3$ propagates $x_2 = 1$
- Conflict-driven search:
 - ▶ Try to build satisfying assignment literal by literal using decisions and propagations
 - ▶ When falsifying constraint, derive constraint explaining the conflict and add to formula

Conflict Analysis Example

- Let

$$C_1 \doteq \bar{z} + \bar{w} \geq 1 \quad C_2 \doteq \bar{y} + w \geq 1 \quad C_3 \doteq 2x + y + z \geq 2$$

Conflict Analysis Example

- Let

$$C_1 \doteq \bar{z} + \bar{w} \geq 1 \quad C_2 \doteq \bar{y} + w \geq 1 \quad C_3 \doteq 2x + y + z \geq 2$$

- Decide $x = 0$

Conflict Analysis Example

- Let

$$C_1 \doteq \bar{z} + \bar{w} \geq 1 \quad C_2 \doteq \bar{y} + w \geq 1 \quad C_3 \doteq 2x + y + z \geq 2$$

- Decide $x = 0$
- C_3 propagates $y = 1$ and $z = 1$

Conflict Analysis Example

- Let

$$C_1 \doteq \bar{z} + \bar{w} \geq 1 \quad C_2 \doteq \bar{y} + w \geq 1 \quad C_3 \doteq 2x + y + z \geq 2$$

- Decide $x = 0$
- C_3 propagates $y = 1$ and $z = 1$
- C_2 propagates $w = 1$

Conflict Analysis Example

- Let

$$C_1 \doteq \bar{z} + \bar{w} \geq 1 \quad C_2 \doteq \bar{y} + w \geq 1 \quad C_3 \doteq 2x + y + z \geq 2$$

- Decide $x = 0$
- C_3 propagates $y = 1$ and $z = 1$
- C_2 propagates $w = 1$
- C_1 is falsified – we found a conflict!

Conflict Analysis Example

- Let

$$C_1 \doteq \bar{z} + \bar{w} \geq 1 \quad C_2 \doteq \bar{y} + w \geq 1 \quad C_3 \doteq 2x + y + z \geq 2$$

- Decide $x = 0$
- C_3 propagates $y = 1$ and $z = 1$
- C_2 propagates $w = 1$
- C_1 is falsified – we found a conflict!

Conflict analysis to learn $x = 1$:

$$\bar{z} + \bar{w} \geq 1 \quad \bar{y} + w \geq 1$$

Conflict Analysis Example

- Let

$$C_1 \doteq \bar{z} + \bar{w} \geq 1 \quad C_2 \doteq \bar{y} + w \geq 1 \quad C_3 \doteq 2x + y + z \geq 2$$

- Decide $x = 0$
- C_3 propagates $y = 1$ and $z = 1$
- C_2 propagates $w = 1$
- C_1 is falsified – we found a conflict!

Conflict analysis to learn $x = 1$:

$$\text{Add } \frac{\bar{z} + \bar{w} \geq 1 \quad \bar{y} + w \geq 1}{\bar{y} + \bar{z} + \bar{w} + w \geq 2}$$

Conflict Analysis Example

- Let

$$C_1 \doteq \bar{z} + \bar{w} \geq 1 \quad C_2 \doteq \bar{y} + w \geq 1 \quad C_3 \doteq 2x + y + z \geq 2$$

- Decide $x = 0$
- C_3 propagates $y = 1$ and $z = 1$
- C_2 propagates $w = 1$
- C_1 is falsified – we found a conflict!

Conflict analysis to learn $x = 1$:

$$\text{Add } \frac{\begin{array}{c} \bar{z} + \bar{w} \geq 1 \\ \bar{y} + w \geq 1 \end{array}}{\bar{y} + \bar{z} \geq 1}$$

Conflict Analysis Example

- Let

$$C_1 \doteq \bar{z} + \bar{w} \geq 1 \quad C_2 \doteq \bar{y} + w \geq 1 \quad C_3 \doteq 2x + y + z \geq 2$$

- Decide $x = 0$
- C_3 propagates $y = 1$ and $z = 1$
- C_2 propagates $w = 1$
- C_1 is falsified – we found a conflict!

Conflict analysis to learn $x = 1$:

$$\text{Add } \frac{\begin{array}{c} \bar{z} + \bar{w} \geq 1 \\ \bar{y} + w \geq 1 \end{array}}{\begin{array}{c} \bar{y} + \bar{z} \\ \geq 1 \end{array}} \quad 2x + y + z \geq 2$$

Conflict Analysis Example

- Let

$$C_1 \doteq \bar{z} + \bar{w} \geq 1 \quad C_2 \doteq \bar{y} + w \geq 1 \quad C_3 \doteq 2x + y + z \geq 2$$

- Decide $x = 0$
- C_3 propagates $y = 1$ and $z = 1$
- C_2 propagates $w = 1$
- C_1 is falsified – we found a conflict!

Conflict analysis to learn $x = 1$:

$$\begin{array}{c} \bar{z} + \bar{w} \geq 1 \quad \bar{y} + w \geq 1 \\ \text{Add} \quad \hline \bar{y} + \bar{z} \quad \geq 1 \quad 2x + y + z \geq 2 \\ \text{Add} \quad \hline 2x \geq 1 \end{array}$$

Conflict Analysis Example

- Let

$$C_1 \doteq \bar{z} + \bar{w} \geq 1 \quad C_2 \doteq \bar{y} + w \geq 1 \quad C_3 \doteq 2x + y + z \geq 2$$

- Decide $x = 0$
- C_3 propagates $y = 1$ and $z = 1$
- C_2 propagates $w = 1$
- C_1 is falsified – we found a conflict!

Conflict analysis to learn $x = 1$:

$$\begin{array}{c}
 \begin{array}{ccc}
 \bar{z} + \bar{w} \geq 1 & \bar{y} + w \geq 1 & \\
 \text{Add} \quad \hline
 \bar{y} + \bar{z} & \geq 1 & 2x + y + z \geq 2
 \end{array} \\
 \text{Add} \quad \hline
 \begin{array}{c}
 \text{Divide by 2} \quad \hline
 x \geq \frac{1}{2}
 \end{array}
 \end{array}$$

Conflict Analysis Example

- Let

$$C_1 \doteq \bar{z} + \bar{w} \geq 1 \quad C_2 \doteq \bar{y} + w \geq 1 \quad C_3 \doteq 2x + y + z \geq 2$$

- Decide $x = 0$
- C_3 propagates $y = 1$ and $z = 1$
- C_2 propagates $w = 1$
- C_1 is falsified – we found a conflict!

Conflict analysis to learn $x = 1$:

$$\begin{array}{c}
 \begin{array}{ccc}
 \bar{z} + \bar{w} \geq 1 & \bar{y} + w \geq 1 & \\
 \text{Add} \quad \hline
 \bar{y} + \bar{z} & \geq 1 & 2x + y + z \geq 2
 \end{array} \\
 \text{Add} \quad \hline
 \begin{array}{c}
 \text{Divide by 2} \quad \hline
 2x \geq 1 \\
 x \geq 1
 \end{array}
 \end{array}$$

Approaches for Pseudo-Boolean Solving and Optimization

- Two main approaches for pseudo-Boolean solving:
 - ▶ CNF-based: Translate to CNF and run conflict-driven clause learning (CDCL)
 - ▶ Native PB: Generalize conflict-driven search to pseudo-Boolean constraints ([focus of this talk](#))

Approaches for Pseudo-Boolean Solving and Optimization

- Two main approaches for pseudo-Boolean solving:
 - ▶ CNF-based: Translate to CNF and run conflict-driven clause learning (CDCL)
 - ▶ Native PB: Generalize conflict-driven search to pseudo-Boolean constraints ([focus of this talk](#))
- New challenges and techniques for native PB solving 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, we can just print the learned clause
(and check that it follows by reverse unit propagation (RUP))
 - ▶ In PB, explicit description of conflict analysis steps required

Proof Logging for Pseudo-Boolean Optimization

- Conflict analysis:
 - ▶ In SAT, we can just print the learned clause
(and check that it follows by reverse unit propagation (RUP))
 - ▶ In PB, explicit description of conflict analysis steps required
- 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, we can just print the learned clause
(and check that it follows by reverse unit propagation (RUP))
 - ▶ In PB, explicit description of conflict analysis steps required
- Other techniques pose further challenges:
 - ▶ Objective rewriting in core-guided search
 - ▶ Linear programming (LP) integration (Farkas certificates, cut generation, ...)
- Low-level challenges for truly efficient proof logging and checking:
 - ▶ Logging unit constraints (saying that a variable must take some fixed value, e.g. $x_2 \geq 1$)
 - ▶ Logging constraint simplifications (e.g. simplifying away variables with fixed values)
 - ▶ Logging and checking solutions
 - ▶ Optimizing 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

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

Addition

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

Pseudo-Boolean Proof Logging Basics

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

Input axioms

From the input

Literal axioms

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

Addition

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

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

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

Addition

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

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 $c \in \mathbb{N}^+$

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

The Division Rule

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

Proof of soundness:

- Dividing $\sum_i a_i \ell_i \geq A$ by c yields $\sum_i \frac{a_i}{c} \ell_i \geq \frac{A}{c}$

The Division Rule

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

Proof of soundness:

- Dividing $\sum_i a_i \ell_i \geq A$ by c yields $\sum_i \frac{a_i}{c} \ell_i \geq \frac{A}{c}$
- Rounding up coefficients on the LHS: $\sum_i \lceil \frac{a_i}{c} \rceil \ell_i \geq \frac{A}{c}$ (valid since $\ell_i \geq 0$)

The Division Rule

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

Proof of soundness:

- Dividing $\sum_i a_i \ell_i \geq A$ by c yields $\sum_i \frac{a_i}{c} \ell_i \geq \frac{A}{c}$
- Rounding up coefficients on the LHS: $\sum_i \lceil \frac{a_i}{c} \rceil \ell_i \geq \frac{A}{c}$ (valid since $\ell_i \geq 0$)
- The LHS is an integer, so can round up RHS to next integer: $\sum_i \lceil \frac{a_i}{c} \rceil \ell_i \geq \left\lceil \frac{A}{c} \right\rceil$

The Division Rule

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

Proof of soundness:

- Dividing $\sum_i a_i \ell_i \geq A$ by c yields $\sum_i \frac{a_i}{c} \ell_i \geq \frac{A}{c}$
- Rounding up coefficients on the LHS: $\sum_i \lceil \frac{a_i}{c} \rceil \ell_i \geq \frac{A}{c}$ (valid since $\ell_i \geq 0$)
- The LHS is an integer, so can round up RHS to next integer: $\sum_i \lceil \frac{a_i}{c} \rceil \ell_i \geq \left\lceil \frac{A}{c} \right\rceil$

Division is crucial for Boolean (as opposed to real-valued) reasoning:

- Addition and multiplication valid over the reals
- Literal axioms $\ell_i \geq 0$ and $\bar{\ell}_i = 1 - \ell_i \geq 0$ valid for all reals in $[0, 1]$
- Division only valid over the integers: e.g. $2x_1 \geq 1$ implies $x_1 \geq 1$

Conflict Analysis Example: VeriPB Derivation

$$\frac{\begin{array}{c} \text{Add } \frac{\bar{z} + \bar{w} \geq 1 \quad \bar{y} + w \geq 1}{\bar{y} + \bar{z} \geq 1} \\ \text{Add } \end{array}}{2x + y + z \geq 2}$$

Divide by 2 $\frac{2x \geq 1}{x \geq 1}$

Conflict Analysis Example: VeriPB Derivation

$$\begin{array}{c}
 \text{Add} \quad \frac{\bar{z} + \bar{w} \geq 1 \quad \bar{y} + w \geq 1}{\bar{y} + \bar{z} \geq 1} \quad 2x + y + z \geq 2 \\
 \text{Add} \quad \frac{\bar{y} + \bar{z} \geq 1}{\text{Divide by 2}} \quad \frac{2x \geq 1}{x \geq 1}
 \end{array}$$

By naming constraints by labels as

$$\text{Constraint } @C1 \doteq \bar{z} + \bar{w} \geq 1$$

$$\text{Constraint } @C2 \doteq \bar{y} + w \geq 1$$

$$\text{Constraint } @C3 \doteq 2x + y + z \geq 2$$

Conflict Analysis Example: VeriPB Derivation

$$\begin{array}{c}
 \text{Add} \quad \frac{\bar{z} + \bar{w} \geq 1 \quad \bar{y} + w \geq 1}{\bar{y} + \bar{z} \geq 1} \quad 2x + y + z \geq 2 \\
 \text{Add} \quad \frac{}{\text{Divide by 2}} \quad \frac{2x \geq 1}{x \geq 1}
 \end{array}$$

By naming constraints by labels as

$$\text{Constraint } @C1 \doteq \bar{z} + \bar{w} \geq 1$$

$$\text{Constraint } @C2 \doteq \bar{y} + w \geq 1$$

$$\text{Constraint } @C3 \doteq 2x + y + z \geq 2$$

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

```
pol  @C1  @C2  +  @C3  +  2  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 $F \cup \{\textcolor{red}{C}\}$ are **equisatisfiable** if there is a substitution ω (mapping variables to truth values or literals), called a **witness**, for which

$$F \cup \{\neg \textcolor{red}{C}\} \models (F \cup \{\textcolor{red}{C}\}) \upharpoonright_{\omega}$$

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

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

When using rule in a proof, the implication needs to be **efficiently verifiable** — every $D \in (F \cup \{C\}) \upharpoonright \omega$ should follow from $F \cup \{\neg C\}$ either “obviously” or by explicit derivation

Redundance-Based Strengthening: Example

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

Redundance-Based Strengthening: Example

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

Redundance-Based Strengthening: Example

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

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

Redundance-Based Strengthening: Example

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

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

Choose $\omega = \{y_3 \mapsto 1\}$ — F untouched; new constraint $C_1 \upharpoonright_{\omega}$ trivially satisfied

Redundance-Based Strengthening: Example

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

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

Choose $\omega = \{y_3 \mapsto 1\}$ — F untouched; new constraint $C_1\upharpoonright_\omega$ trivially satisfied

- $F \cup \{C_1\} \cup \{\neg C_2\} \models (F \cup \{C_1\} \cup \{C_2\})\upharpoonright_\omega$

Redundance-Based Strengthening: Example

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

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

Choose $\omega = \{y_3 \mapsto 1\}$ — F untouched; new constraint $C_1\upharpoonright_\omega$ trivially satisfied

- $F \cup \{C_1\} \cup \{\neg C_2\} \models (F \cup \{C_1\} \cup \{C_2\})\upharpoonright_\omega$

Choose $\omega = \{y_3 \mapsto 0\}$ — F untouched; new constraint $C_2\upharpoonright_\omega$ follows from D ;
 $\neg C_2 \doteq x_1 + x_2 + x_3 \leq 1 + y_3$ implies $C_1\upharpoonright_\omega \doteq x_1 + x_2 + x_3 \leq 2$

Redundance-Based Strengthening: Example

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

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

Choose $\omega = \{y_3 \mapsto 1\}$ — F untouched; new constraint $C_1\upharpoonright_\omega$ trivially satisfied

- $F \cup \{C_1\} \cup \{\neg C_2\} \models (F \cup \{C_1\} \cup \{C_2\})\upharpoonright_\omega$

Choose $\omega = \{y_3 \mapsto 0\}$ — F untouched; new constraint $C_2\upharpoonright_\omega$ follows from D ;

$\neg C_2 \doteq x_1 + x_2 + x_3 \leq 1 + y_3$ implies $C_1\upharpoonright_\omega \doteq x_1 + x_2 + x_3 \leq 2$

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;

Proof by Contradiction

- F and $F \cup \{C\}$ are equisatisfiable if $F \cup \{\neg C\} \models \perp$
- Can be seen as a special case of the redundancy rule (empty witness w)

Proof by Contradiction: Example

- From

$$C_1 \doteq 2t + x_1 + x_2 \geq 2 \quad C_2 \doteq 2\bar{t} + x_1 + x_2 \geq 2$$

derive $D \doteq x_1 + x_2 \geq 2$

Proof by Contradiction: Example

- From

$$C_1 \doteq 2t + x_1 + x_2 \geq 2 \quad C_2 \doteq 2\bar{t} + x_1 + x_2 \geq 2$$

derive $D \doteq x_1 + x_2 \geq 2$

- Can use negation $\neg D \doteq x_1 + x_2 \leq 1 \doteq -x_1 - x_2 \geq -1$

Proof by Contradiction: Example

- From

$$C_1 \doteq 2t + x_1 + x_2 \geq 2 \quad C_2 \doteq 2\bar{t} + x_1 + x_2 \geq 2$$

derive $D \doteq x_1 + x_2 \geq 2$

- Can use negation $\neg D \doteq x_1 + x_2 \leq 1 \doteq -x_1 - x_2 \geq -1$

$$2t + x_1 + x_2 \geq 2 \quad -x_1 - x_2 \geq -1$$

Proof by Contradiction: Example

- From

$$C_1 \doteq 2t + x_1 + x_2 \geq 2 \quad C_2 \doteq 2\bar{t} + x_1 + x_2 \geq 2$$

derive $D \doteq x_1 + x_2 \geq 2$

- Can use negation $\neg D \doteq x_1 + x_2 \leq 1 \doteq -x_1 - x_2 \geq -1$

Add
$$\frac{2t + x_1 + x_2 \geq 2 \quad -x_1 - x_2 \geq -1}{2t \geq 1}$$

Proof by Contradiction: Example

- From

$$C_1 \doteq 2t + x_1 + x_2 \geq 2 \quad C_2 \doteq 2\bar{t} + x_1 + x_2 \geq 2$$

derive $D \doteq x_1 + x_2 \geq 2$

- Can use negation $\neg D \doteq x_1 + x_2 \leq 1 \doteq -x_1 - x_2 \geq -1$

$$\text{Add } \frac{2t + x_1 + x_2 \geq 2 \quad -x_1 - x_2 \geq -1}{2t \geq 1}$$

Divide by 2 $t \geq 1$

Proof by Contradiction: Example

- From

$$C_1 \doteq 2t + x_1 + x_2 \geq 2 \quad C_2 \doteq 2\bar{t} + x_1 + x_2 \geq 2$$

derive $D \doteq x_1 + x_2 \geq 2$

- Can use negation $\neg D \doteq x_1 + x_2 \leq 1 \doteq -x_1 - x_2 \geq -1$

$$\text{Add } \frac{2t + x_1 + x_2 \geq 2 \quad -x_1 - x_2 \geq -1}{\begin{array}{c} 2t \geq 1 \\ \hline t \geq 1 \end{array}}$$

Divide by 2

$$\text{Add } \frac{2\bar{t} + x_1 + x_2 \geq 2 \quad -x_1 - x_2 \geq -1}{\begin{array}{c} 2\bar{t} \geq 1 \\ \hline \bar{t} \geq 1 \end{array}}$$

Divide by 2

Proof by Contradiction: Example

- From

$$C_1 \doteq 2t + x_1 + x_2 \geq 2 \quad C_2 \doteq 2\bar{t} + x_1 + x_2 \geq 2$$

derive $D \doteq x_1 + x_2 \geq 2$

- Can use negation $\neg D \doteq x_1 + x_2 \leq 1 \doteq -x_1 - x_2 \geq -1$

$$\begin{array}{c}
 \text{Add} \quad \dfrac{2t + x_1 + x_2 \geq 2 \quad -x_1 - x_2 \geq -1}{2t \geq 1} \quad \text{Add} \quad \dfrac{2\bar{t} + x_1 + x_2 \geq 2 \quad -x_1 - x_2 \geq -1}{2\bar{t} \geq 1} \\
 \text{Divide by 2} \quad \dfrac{2t \geq 1}{t \geq 1} \quad \text{Divide by 2} \quad \dfrac{2\bar{t} \geq 1}{\bar{t} \geq 1} \\
 \text{Add} \quad \dfrac{}{0 \geq 1}
 \end{array}$$

Proof by Contradiction: Example

- From

$$C_1 \doteq 2t + x_1 + x_2 \geq 2 \quad C_2 \doteq 2\bar{t} + x_1 + x_2 \geq 2$$

derive $D \doteq x_1 + x_2 \geq 2$

- Can use negation $\neg D \doteq x_1 + x_2 \leq 1 \doteq -x_1 - x_2 \geq -1$

$$\begin{array}{c}
 \text{Add } \frac{2t + x_1 + x_2 \geq 2 \quad -x_1 - x_2 \geq -1}{2t \geq 1} \quad \text{Add } \frac{2\bar{t} + x_1 + x_2 \geq 2 \quad -x_1 - x_2 \geq -1}{2\bar{t} \geq 1} \\
 \text{Divide by 2} \quad \frac{2t \geq 1}{t \geq 1} \quad \text{Divide by 2} \quad \frac{2\bar{t} \geq 1}{\bar{t} \geq 1} \\
 \text{Add } \frac{}{0 \geq 1}
 \end{array}$$

```
pbc +1 x1 +1 x2 >= 2 : subproof
VeriPB:      pol @C1 -1 + 2 d @C2 -1 + 2 d + ;
            qed pbc : -1;
```

Proof Logging for Decision and Optimization Problems

- Decision problems:
 - ▶ Satisfiable instances: just provide a solution
 - ▶ Unsatisfiable instances: derivation of contradiction $0 \geq 1$

Proof Logging for Decision and Optimization Problems

- Decision problems:
 - ▶ Satisfiable instances: just provide a solution
 - ▶ Unsatisfiable instances: derivation of contradiction $0 \geq 1$
- Optimization problems: provide:
 - (i) a solution with value UB , and
 - (ii) a derivation of the inequality $Obj \geq LB$

(Optimality proven if $UB = LB$)

Solution-Improving Search

- Find solutions with better and better objective values
- When finding solution with value v , introduce **objective-improving constraint** $Obj \leq v - 1$

Solution-Improving Search

- Find solutions with better and better objective values
- When finding solution with value v , introduce **objective-improving constraint** $Obj \leq v - 1$
- After finding optimal solution with value v^* , derive contradiction from $Obj \leq v^* - 1$

Solution-Improving Search

- Find solutions with better and better objective values
- When finding solution with value v , introduce **objective-improving constraint** $\text{Obj} \leq v - 1$
- After finding optimal solution with value v^* , derive contradiction from $\text{Obj} \leq v^* - 1$
- Proof logging:
 - ▶ Objective-improving constraints are provided by the `soli` rule in *VeriPB*
 - ▶ Final contradiction implies $\text{Obj} \geq v^*$

Solution-Improving Search

- Find solutions with better and better objective values
- When finding solution with value v , introduce **objective-improving constraint** $Obj \leq v - 1$
- After finding optimal solution with value v^* , derive contradiction from $Obj \leq v^* - 1$
- Proof logging:
 - ▶ Objective-improving constraints are provided by the `soli` rule in *VeriPB*
 - ▶ Final contradiction implies $Obj \geq v^*$
- Example: Let $Obj = x_1 + 2x_2 + x_3$
We find the solution $x_1 = x_3 = 1, x_2 = 0$ with objective value 2
Then `soli x1 ~x2 x3` introduces constraint $Obj \leq 1$, i.e. $x_1 + 2x_2 + x_3 \leq 1$

Running Decision Solver with Assumptions

- Recall: conflict-driven search tries to build satisfying assignment
- Can also do this starting from pre-chosen literal values
- These pre-chosen values are called **assumptions**

Running Decision Solver with Assumptions

- Recall: conflict-driven search tries to build satisfying assignment
- Can also do this starting from pre-chosen literal values
- These pre-chosen values are called **assumptions**
- Possible outcomes:
 - ▶ Consistent \implies find solution to formula
 - ▶ Inconsistent \implies learn constraint (called **core**) why assumptions are inconsistent

Core-Guided Optimization

- Make most optimistic assumption: all literals ℓ_i in objective are 0

Core-Guided Optimization

- Make most optimistic assumption: all literals ℓ_i in objective are 0
- If assumptions consistent: found **optimal** solution

Core-Guided Optimization

- Make most optimistic assumption: all literals ℓ_i in objective are 0
- If assumptions consistent: found **optimal** solution
- If assumptions inconsistent: derive **core constraint**

$$\sum_{i=1}^k \ell_i \geq A$$

Core-Guided Optimization

- Make most optimistic assumption: all literals ℓ_i in objective are 0
- If assumptions consistent: found **optimal** solution
- If assumptions inconsistent: derive **core constraint**

$$\sum_{i=1}^k \ell_i \geq A$$

- Introduce fresh variables y_k such that

$$\sum_{i=1}^k \ell_i = A + \sum_{i=A+1}^k y_i$$

$(y_j$ is true iff $\sum_{i=1}^k \ell_i \geq j$ for $A+1 \leq j \leq k)$

Core-Guided Optimization

- Make most optimistic assumption: all literals ℓ_i in objective are 0
- If assumptions consistent: found **optimal** solution
- If assumptions inconsistent: derive **core constraint**

$$\sum_{i=1}^k \ell_i \geq A$$

- Introduce fresh variables y_k such that

$$\sum_{i=1}^k \ell_i = A + \sum_{i=A+1}^k y_i$$

$(y_j$ is true iff $\sum_{i=1}^k \ell_i \geq j$ for $A+1 \leq j \leq k)$

- Use this equality to cancel literal with lowest coefficient from objective

Core-Guided Optimization

- Make most optimistic assumption: all literals ℓ_i in objective are 0
- If assumptions consistent: found **optimal** solution
- If assumptions inconsistent: derive **core constraint**

$$\sum_{i=1}^k \ell_i \geq A$$

- Introduce fresh variables y_k such that

$$\sum_{i=1}^k \ell_i = A + \sum_{i=A+1}^k y_i$$

$(y_j$ is true iff $\sum_{i=1}^k \ell_i \geq j$ for $A+1 \leq j \leq k)$

- Use this equality to cancel literal with lowest coefficient from objective
- Repeat with rewritten objective

Core-Guided Optimization: Example

- Objective:

$$Obj = x_1 + 2x_2 + 3x_3 + 4x_4$$

Core-Guided Optimization: Example

- Objective:

$$Obj = x_1 + 2x_2 + 3x_3 + 4x_4$$

- Assume

$$x_1 = x_2 = x_3 = x_4 = 0$$

Core-Guided Optimization: Example

- Objective:

$$Obj = x_1 + 2x_2 + 3x_3 + 4x_4$$

- Assume

$$x_1 = x_2 = x_3 = x_4 = 0$$

- Decision solver: Inconsistent, core constraint:

$$x_2 + x_3 + x_4 \geq 2$$

Core-Guided Optimization: Example

- Objective:

$$Obj = x_1 + 2x_2 + 3x_3 + 4x_4$$

- Assume

$$x_1 = x_2 = x_3 = x_4 = 0$$

- Decision solver: Inconsistent, core constraint:

$$x_2 + x_3 + x_4 \geq 2$$

- Introduce variable y_3 such that

$$x_2 + x_3 + x_4 = 2 + y_3$$

Core-Guided Optimization: Example

- Objective:

$$Obj = x_1 + 2x_2 + 3x_3 + 4x_4$$

- Assume

$$x_1 = x_2 = x_3 = x_4 = 0$$

- Decision solver: Inconsistent, core constraint:

$$x_2 + x_3 + x_4 \geq 2$$

- Introduce variable y_3 such that

$$x_2 + x_3 + x_4 = 2 + y_3$$

- Rewrite objective:

$$\begin{aligned} Obj &= x_1 + 2(x_2 + x_3 + x_4) + x_3 + 2x_4 \\ &= x_1 + 2(2 + y_3) + x_3 + 2x_4 \\ &= x_1 + x_3 + 2x_4 + 2y_3 + 4 \end{aligned}$$

Core-Guided Optimization: Example

- Objective:

$$Obj = x_1 + 2x_2 + 3x_3 + 4x_4$$

- Assume

$$x_1 = x_2 = x_3 = x_4 = 0$$

- Decision solver: Inconsistent, core constraint:

$$x_2 + x_3 + x_4 \geq 2$$

- Introduce variable y_3 such that

$$x_2 + x_3 + x_4 = 2 + y_3$$

- Rewrite objective:

$$\begin{aligned} Obj &= x_1 + 2(x_2 + x_3 + x_4) + x_3 + 2x_4 \\ &= x_1 + 2(2 + y_3) + x_3 + 2x_4 \\ &= x_1 + x_3 + 2x_4 + 2y_3 + 4 \end{aligned}$$

- Shows that $Obj \geq 4$

Core-Guided Optimization: Example

- Objective:

$$Obj = x_1 + 2x_2 + 3x_3 + 4x_4$$

- Assume

$$x_1 = x_2 = x_3 = x_4 = 0$$

- Decision solver: Inconsistent, core constraint:

$$x_2 + x_3 + x_4 \geq 2$$

- Introduce variable y_3 such that

$$x_2 + x_3 + x_4 = 2 + y_3$$

- Rewrite objective:

$$\begin{aligned} Obj &= x_1 + 2(x_2 + x_3 + x_4) + x_3 + 2x_4 \\ &= x_1 + 2(2 + y_3) + x_3 + 2x_4 \\ &= x_1 + x_3 + 2x_4 + 2y_3 + 4 \end{aligned}$$

- Shows that $Obj \geq 4$

- Next assume $x_1 = x_3 = x_4 = y_3 = 0 \dots$

Proof Logging for Core-Guided Optimization: Example

We know $D \doteq x_2 + x_3 + x_4 \geq 2$. Want to introduce a variable y_3 such that

$$x_2 + x_3 + x_4 = 2 + y_3, \quad \text{i.e.} \quad \begin{cases} C_1 & \doteq x_2 + x_3 + x_4 \leq 2 + y_3 \\ C_2 & \doteq x_2 + x_3 + x_4 \geq 2 + y_3 \end{cases}$$

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

Proof Logging for Core-Guided Optimization: Example

We know $D \doteq x_2 + x_3 + x_4 \geq 2$. Want to introduce a variable y_3 such that

$$x_2 + x_3 + x_4 = 2 + y_3, \quad \text{i.e.} \quad \begin{cases} C_1 & \doteq x_2 + x_3 + x_4 \leq 2 + y_3 \\ C_2 & \doteq x_2 + x_3 + x_4 \geq 2 + y_3 \end{cases}$$

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

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

Proof Logging for Core-Guided Optimization: Example

We know $D \doteq x_2 + x_3 + x_4 \geq 2$. Want to introduce a variable y_3 such that

$$x_2 + x_3 + x_4 = 2 + y_3, \quad \text{i.e.} \quad \begin{cases} C_1 & \doteq x_2 + x_3 + x_4 \leq 2 + y_3 \\ C_2 & \doteq x_2 + x_3 + x_4 \geq 2 + y_3 \end{cases}$$

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

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

Choose $\omega = \{y_3 \mapsto 1\}$ — F untouched; new constraint $C_1 \upharpoonright_{\omega}$ trivially satisfied

Proof Logging for Core-Guided Optimization: Example

We know $D \doteq x_2 + x_3 + x_4 \geq 2$. Want to introduce a variable y_3 such that

$$x_2 + x_3 + x_4 = 2 + y_3, \quad \text{i.e.} \quad \begin{cases} C_1 & \doteq x_2 + x_3 + x_4 \leq 2 + y_3 \\ C_2 & \doteq x_2 + x_3 + x_4 \geq 2 + y_3 \end{cases}$$

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

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

Choose $\omega = \{y_3 \mapsto 1\}$ — F untouched; new constraint $C_1 \upharpoonright_{\omega}$ trivially satisfied

- $F \cup \{C_1\} \cup \{\neg C_2\} \models (F \cup \{C_1\} \cup \{C_2\}) \upharpoonright_{\omega}$

Proof Logging for Core-Guided Optimization: Example

We know $D \doteq x_2 + x_3 + x_4 \geq 2$. Want to introduce a variable y_3 such that

$$x_2 + x_3 + x_4 = 2 + y_3, \quad \text{i.e.} \quad \begin{cases} C_1 & \doteq x_2 + x_3 + x_4 \leq 2 + y_3 \\ C_2 & \doteq x_2 + x_3 + x_4 \geq 2 + y_3 \end{cases}$$

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

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

Choose $\omega = \{y_3 \mapsto 1\}$ — F untouched; new constraint $C_1 \upharpoonright_{\omega}$ trivially satisfied

- $F \cup \{C_1\} \cup \{\neg C_2\} \models (F \cup \{C_1\} \cup \{C_2\}) \upharpoonright_{\omega}$

Choose $\omega = \{y_3 \mapsto 0\}$ — F untouched; new constraint $C_2 \upharpoonright_{\omega}$ follows from D ;

$\neg C_2 \doteq x_2 + x_3 + x_4 \leq 1 + y_3$ implies $C_1 \upharpoonright_{\omega} \doteq x_2 + x_3 + x_4 \leq 2$

Proof Logging for Core-Guided Optimization: Example

We know $D \doteq x_2 + x_3 + x_4 \geq 2$. Want to introduce a variable y_3 such that

$$x_2 + x_3 + x_4 = 2 + y_3, \quad \text{i.e.} \quad \begin{cases} C_1 & \doteq x_2 + x_3 + x_4 \leq 2 + y_3 \\ C_2 & \doteq x_2 + x_3 + x_4 \geq 2 + y_3 \end{cases}$$

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

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

Choose $\omega = \{y_3 \mapsto 1\}$ — F untouched; new constraint $C_1 \upharpoonright_{\omega}$ trivially satisfied

- $F \cup \{C_1\} \cup \{\neg C_2\} \models (F \cup \{C_1\} \cup \{C_2\}) \upharpoonright_{\omega}$

Choose $\omega = \{y_3 \mapsto 0\}$ — F untouched; new constraint $C_2 \upharpoonright_{\omega}$ follows from D ;

$\neg C_2 \doteq x_2 + x_3 + x_4 \leq 1 + y_3$ implies $C_1 \upharpoonright_{\omega} \doteq x_2 + x_3 + x_4 \leq 2$

VeriPB:

red +1 x2 +1 x3 +1 x4 -1 y3 <= 2 : y3 -> 1;
red +1 x2 +1 x3 +1 x4 -1 y3 >= 2 : y3 -> 0;

Proof Logging for Core-Guided Optimization: Some Further Details

$$Obj_{\text{orig}} = x_1 + 2(\textcolor{blue}{x_2 + x_3 + x_4}) + x_3 + 2x_4$$

$$Obj_{\text{rewritten}} = x_1 + 2(\textcolor{green}{2 + y_3}) + x_3 + 2x_4$$

- Multiplying $x_2 + x_3 + x_4 \geq \textcolor{green}{2 + y_3}$ by 2 yields inequality $Obj_{\text{orig}} \geq Obj_{\text{rewritten}}$ (after canceling rest of objective from both sides)
- Used to show, e.g., that $Obj_{\text{rewritten}} \geq LB$ implies $Obj_{\text{orig}} \geq LB$
- Other inequality needed in solver

LP Relaxation

- Linear programming (LP) relaxation: allow variables to take any real value in $[0, 1]$
- In practice usually solved quickly using simplex algorithm
- Relaxation has a better/lower optimal objective value

Pseudo-Boolean Solving: LP Integration

- Recall: conflict-driven search tries to build satisfying assignment
- Partial assignments may yield unsatisfiable subproblem even over the reals
- Propagation does not necessarily detect this, but LP solving can

Pseudo-Boolean Solving: LP Integration

- Recall: conflict-driven search tries to build satisfying assignment
- Partial assignments may yield unsatisfiable subproblem even over the reals
- Propagation does not necessarily detect this, but LP solving can
- Possible outcomes when solving LP relaxation on formula + partial assignment:
 - ▶ infeasibility \implies generate Farkas certificate
 - ▶ found integral solution \implies this solution is optimal
 - ▶ found fractional solution \implies add constraints ‘cutting away’ fractional solution:
cut generation

Farkas Certificates

If solver decides $y = 0$, then constraints

$$C_1 \doteq y + x_1 + x_2 + x_3 \geq 2$$

$$C_2 \doteq y + 3x_1 + 2x_2 + x_3 + x_4 \geq 3$$

$$C_3 \doteq -2x_1 - 2x_2 - x_3 \geq -1$$

are infeasible over the reals, so $y \geq 1$ must hold

Farkas Certificates

If solver decides $y = 0$, then constraints

$$C_1 \doteq y + x_1 + x_2 + x_3 \geq 2$$

$$C_2 \doteq y + 3x_1 + 2x_2 + x_3 + x_4 \geq 3$$

$$C_3 \doteq -2x_1 - 2x_2 - x_3 \geq -1$$

are infeasible over the reals, so $y \geq 1$ must hold

LP solver can detect this, but we cannot trust its floating-point arithmetic...

Farkas Certificates

If solver decides $y = 0$, then constraints

$$\textcolor{red}{C}_1 \doteq y + x_1 + x_2 + x_3 \geq 2$$

$$\textcolor{red}{C}_2 \doteq y + 3x_1 + 2x_2 + x_3 + x_4 \geq 3$$

$$\textcolor{red}{C}_3 \doteq -2x_1 - 2x_2 - x_3 \geq -1$$

are infeasible over the reals, so $y \geq 1$ must hold

LP solver can detect this, but we cannot trust its floating-point arithmetic...

Solution: Ask LP solver for **Farkas certificate**: positive linear combination of constraints (and literal axioms, e.g. $\bar{x}_4 \geq 0 \doteq -x_4 \geq -1$) proving infeasibility

Farkas Certificates

If solver decides $y = 0$, then constraints

$$C_1 \doteq y + x_1 + x_2 + x_3 \geq 2$$

$$C_2 \doteq y + 3x_1 + 2x_2 + x_3 + x_4 \geq 3$$

$$C_3 \doteq -2x_1 - 2x_2 - x_3 \geq -1$$

are infeasible over the reals, so $y \geq 1$ must hold

LP solver can detect this, but we cannot trust its floating-point arithmetic...

Solution: Ask LP solver for **Farkas certificate**: positive linear combination of constraints (and literal axioms, e.g. $\bar{x}_4 \geq 0 \doteq -x_4 \geq -1$) proving infeasibility

Round multipliers provided by LP solver to integers and check in exact arithmetic

Farkas Certificates: Proof Logging

For

$$C_1 \doteq y + x_1 + x_2 + x_3 \geq 2$$

$$C_2 \doteq y + 3x_1 + 2x_2 + x_3 + x_4 \geq 3$$

$$C_3 \doteq -2x_1 - 2x_2 - x_3 \geq -1$$

a Farkas certificate is

$$C_1 + C_2 + 2 \cdot C_3 + (\bar{x}_4 \geq 0) + (x_2 \geq 0) \doteq 2y \geq 2$$

Divide by 2 to get $y \geq 1$

Farkas Certificates: Proof Logging

For

$$C_1 \doteq y + x_1 + x_2 + x_3 \geq 2$$

$$C_2 \doteq y + 3x_1 + 2x_2 + x_3 + x_4 \geq 3$$

$$C_3 \doteq -2x_1 - 2x_2 - x_3 \geq -1$$

a Farkas certificate is

$$C_1 + C_2 + 2 \cdot C_3 + (\bar{x}_4 \geq 0) + (x_2 \geq 0) \doteq 2y \geq 2$$

Divide by 2 to get $y \geq 1$

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

Cut Generation: Basics

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

Cut Generation: Basics

- Cut generation:
 - ▶ 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 $x_1 = x_2 = x_3 = \frac{1}{2}$

Cut Generation: Basics

- Cut generation:
 - ▶ 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 $2x_1 + 2x_2 + 2x_3 \geq 3$

Cut Generation: Basics

- Cut generation:
 - ▶ 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 $x_1 + x_2 + x_3 \geq 2$

Cut Generation: Basics

- Cut generation:
 - ▶ 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
- MIR rule is stronger than cutting planes division
- Reasoning uses integer slack variables (not supported by *VeriPB*)
- Proof logging instead uses **proof by contradiction**
- We illustrate this using a concrete example — same method works in general

Advanced Cut Generation: MIR cut

- MIR cut: given a constraint $\sum_i a_i \ell_i \geq A$ and a divisor $d \in \mathbb{N}^+$, derive

$$\sum_i \left(\min \{a_i \bmod d, A \bmod d\} + \left\lfloor \frac{a_i}{d} \right\rfloor (A \bmod d) \right) \ell_i \geq \left\lfloor \frac{A}{d} \right\rfloor (A \bmod d)$$

- We call $R = A \bmod d$ the multiplier of the MIR cut
- Example: Applying a MIR cut with divisor $d = 5$ to

$$10x_1 + 5x_2 + 6x_3 + 3x_4 + x_5 \geq 12$$

yields

$$4x_1 + 2x_2 + 3x_3 + 2x_4 + x_5 \geq 6$$

- Cutting planes division by $d = 5$ and multiplying by $R = 12 \bmod 5 = 2$ yields weaker constraint

$$4x_1 + 2x_2 + 4x_3 + 2x_4 + 2x_5 \geq 6$$

Advanced Cut Generation: Example

- For constraints

$$C_1 \doteq 6x_1 + 5x_2 + 2x_3 + 3x_4 \geq 8, \quad C_2 \doteq x_1 + x_3 \geq 1$$

introduce integral slack variables $s_1, s_2 \geq 0$ to obtain

$$C'_1 \doteq 6x_1 + 5x_2 + 2x_3 + 3x_4 - s_1 = 8, \quad C'_2 \doteq x_1 + x_3 - s_2 = 1$$

Advanced Cut Generation: Example

- For constraints

$$C_1 \doteq 6x_1 + 5x_2 + 2x_3 + 3x_4 \geq 8, \quad C_2 \doteq x_1 + x_3 \geq 1$$

introduce integral slack variables $s_1, s_2 \geq 0$ to obtain

$$C'_1 \doteq 6x_1 + 5x_2 + 2x_3 + 3x_4 - s_1 = 8, \quad C'_2 \doteq x_1 + x_3 - s_2 = 1$$

- Compute linear combination $C'_1 + 4 \cdot C'_2$, and only keep \geq part:

$$10x_1 + 5x_2 + 6x_3 + 3x_4 - s_1 - 4s_2 \geq 12$$

Advanced Cut Generation: Example

- For constraints

$$C_1 \doteq 6x_1 + 5x_2 + 2x_3 + 3x_4 \geq 8, \quad C_2 \doteq x_1 + x_3 \geq 1$$

introduce integral slack variables $s_1, s_2 \geq 0$ to obtain

$$C'_1 \doteq 6x_1 + 5x_2 + 2x_3 + 3x_4 - s_1 = 8, \quad C'_2 \doteq x_1 + x_3 - s_2 = 1$$

- Compute linear combination $C'_1 + 4 \cdot C'_2$, and only keep \geq part:

$$10x_1 + 5x_2 + 6x_3 + 3x_4 - s_1 - 4s_2 \geq 12$$

- Apply a MIR cut with divisor $d = 5$ (multiplier $R = 12 \bmod 5 = 2$):

$$4x_1 + 2x_2 + 3x_3 + 2x_4 - s_2 \geq 6$$

Advanced Cut Generation: Example

- For constraints

$$C_1 \doteq 6x_1 + 5x_2 + 2x_3 + 3x_4 \geq 8, \quad C_2 \doteq x_1 + x_3 \geq 1$$

introduce integral slack variables $s_1, s_2 \geq 0$ to obtain

$$C'_1 \doteq 6x_1 + 5x_2 + 2x_3 + 3x_4 - s_1 = 8, \quad C'_2 \doteq x_1 + x_3 - s_2 = 1$$

- Compute linear combination $C'_1 + 4 \cdot C'_2$, and only keep \geq part:

$$10x_1 + 5x_2 + 6x_3 + 3x_4 - s_1 - 4s_2 \geq 12$$

- Apply a MIR cut with divisor $d = 5$ (multiplier $R = 12 \bmod 5 = 2$):

$$4x_1 + 2x_2 + 3x_3 + 2x_4 - s_2 \geq 6$$

- Subtract C'_2 to obtain

$$3x_1 + 2x_2 + 2x_3 + 2x_4 \geq 5$$

Proof Logging for Advanced Cut Generation: Example

$$C_1 \doteq 6x_1 + 5x_2 + 2x_3 + 3x_4 \geq 8, \quad C_2 \doteq x_1 + x_3 \geq 1$$

- We prove resulting cut $D \doteq 3x_1 + 2x_2 + 2x_3 + 2x_4 \geq 5$ by contradiction
- Can use negation $\neg D \doteq 3x_1 + 2x_2 + 2x_3 + 2x_4 \leq 4 \doteq -3x_1 - 2x_2 - 2x_3 - 2x_4 \geq -4$

Proof Logging for Advanced Cut Generation: Example

$$C_1 \doteq 6x_1 + 5x_2 + 2x_3 + 3x_4 \geq 8, \quad C_2 \doteq x_1 + x_3 \geq 1$$

- We prove resulting cut $D \doteq 3x_1 + 2x_2 + 2x_3 + 2x_4 \geq 5$ by contradiction
- Can use negation $\neg D \doteq 3x_1 + 2x_2 + 2x_3 + 2x_4 \leq 4 \doteq -3x_1 - 2x_2 - 2x_3 - 2x_4 \geq -4$

$$-3x_1 - 2x_2 - 2x_3 - 2x_4 \geq -4 \quad 6x_1 + 5x_2 + 2x_3 + 3x_4 \geq 8$$

Proof Logging for Advanced Cut Generation: Example

$$C_1 \doteq 6x_1 + 5x_2 + 2x_3 + 3x_4 \geq 8, \quad C_2 \doteq x_1 + x_3 \geq 1$$

- We prove resulting cut $D \doteq 3x_1 + 2x_2 + 2x_3 + 2x_4 \geq 5$ by contradiction
- Can use negation $\neg D \doteq 3x_1 + 2x_2 + 2x_3 + 2x_4 \leq 4 \doteq -3x_1 - 2x_2 - 2x_3 - 2x_4 \geq -4$

Add
$$\frac{-3x_1 - 2x_2 - 2x_3 - 2x_4 \geq -4 \quad 6x_1 + 5x_2 + 2x_3 + 3x_4 \geq 8}{3x_1 + 3x_2 + x_4 \geq 4}$$

Proof Logging for Advanced Cut Generation: Example

$$C_1 \doteq 6x_1 + 5x_2 + 2x_3 + 3x_4 \geq 8, \quad C_2 \doteq x_1 + x_3 \geq 1$$

- We prove resulting cut $D \doteq 3x_1 + 2x_2 + 2x_3 + 2x_4 \geq 5$ by contradiction
- Can use negation $\neg D \doteq 3x_1 + 2x_2 + 2x_3 + 2x_4 \leq 4 \doteq -3x_1 - 2x_2 - 2x_3 - 2x_4 \geq -4$

Add
$$\frac{-3x_1 - 2x_2 - 2x_3 - 2x_4 \geq -4 \quad 6x_1 + 5x_2 + 2x_3 + 3x_4 \geq 8}{\begin{array}{c} \text{Divide by 3} \\ \hline 3x_1 + 3x_2 + x_4 \geq 4 \\ x_1 + x_2 + x_4 \geq 2 \end{array}}$$

Proof Logging for Advanced Cut Generation: Example

$$\textcolor{red}{C}_1 \doteq 6x_1 + 5x_2 + 2x_3 + 3x_4 \geq 8, \quad \textcolor{red}{C}_2 \doteq x_1 + x_3 \geq 1$$

- We prove resulting cut $\textcolor{blue}{D} \doteq 3x_1 + 2x_2 + 2x_3 + 2x_4 \geq 5$ by contradiction
- Can use negation $\neg\textcolor{blue}{D} \doteq 3x_1 + 2x_2 + 2x_3 + 2x_4 \leq 4 \doteq -3x_1 - 2x_2 - 2x_3 - 2x_4 \geq -4$

$$\begin{array}{r}
 \text{Add } \frac{-3x_1 - 2x_2 - 2x_3 - 2x_4 \geq -4 \quad 6x_1 + 5x_2 + 2x_3 + 3x_4 \geq 8}{3x_1 + 3x_2 + x_4 \geq 4} \\
 \text{Divide by 3 } \frac{3x_1 + 3x_2 + x_4 \geq 4}{x_1 + x_2 + x_4 \geq 2} \\
 \text{Multiply by 2 } \frac{x_1 + x_2 + x_4 \geq 2}{2x_1 + 2x_2 + 2x_4 \geq 4}
 \end{array}$$

Proof Logging for Advanced Cut Generation: Example

$$C_1 \doteq 6x_1 + 5x_2 + 2x_3 + 3x_4 \geq 8, \quad C_2 \doteq x_1 + x_3 \geq 1$$

- We prove resulting cut $D \doteq 3x_1 + 2x_2 + 2x_3 + 2x_4 \geq 5$ by contradiction
- Can use negation $\neg D \doteq 3x_1 + 2x_2 + 2x_3 + 2x_4 \leq 4 \doteq -3x_1 - 2x_2 - 2x_3 - 2x_4 \geq -4$

$$\begin{array}{r}
 \text{Add} \quad \frac{-3x_1 - 2x_2 - 2x_3 - 2x_4 \geq -4 \quad 6x_1 + 5x_2 + 2x_3 + 3x_4 \geq 8}{3x_1 + 3x_2 + x_4 \geq 4} \\
 \text{Divide by 3} \quad \frac{3x_1 + 3x_2 + x_4 \geq 4}{x_1 + x_2 + x_4 \geq 2} \\
 \text{Multiply by 2} \quad \frac{x_1 + x_2 + x_4 \geq 2}{2x_1 + 2x_2 + 2x_4 \geq 4} \\
 \text{Add} \quad \frac{2x_1 + 2x_2 + 2x_4 \geq 4 \quad -3x_1 - 2x_2 - 2x_3 - 2x_4 \geq -4}{-x_1 - 2x_3 \geq 0}
 \end{array}$$

Proof Logging for Advanced Cut Generation: Example

$$C_1 \doteq 6x_1 + 5x_2 + 2x_3 + 3x_4 \geq 8, \quad C_2 \doteq x_1 + x_3 \geq 1$$

- We prove resulting cut $D \doteq 3x_1 + 2x_2 + 2x_3 + 2x_4 \geq 5$ by contradiction
- Can use negation $\neg D \doteq 3x_1 + 2x_2 + 2x_3 + 2x_4 \leq 4 \doteq -3x_1 - 2x_2 - 2x_3 - 2x_4 \geq -4$

$$\begin{array}{r}
 \text{Add} \quad \frac{-3x_1 - 2x_2 - 2x_3 - 2x_4 \geq -4 \quad 6x_1 + 5x_2 + 2x_3 + 3x_4 \geq 8}{3x_1 + 3x_2 + x_4 \geq 4} \\
 \text{Divide by 3} \quad \frac{3x_1 + 3x_2 + x_4 \geq 4}{x_1 + x_2 + x_4 \geq 2} \\
 \text{Multiply by 2} \quad \frac{x_1 + x_2 + x_4 \geq 2}{2x_1 + 2x_2 + 2x_4 \geq 4} \\
 \text{Add} \quad \frac{2x_1 + 2x_2 + 2x_4 \geq 4 \quad -3x_1 - 2x_2 - 2x_3 - 2x_4 \geq -4}{-x_1 - 2x_3 \geq 0} \\
 \text{Add} \quad \frac{-x_1 - 2x_3 \geq 0 \quad x_1 + x_3 \geq 1}{-x_3 \geq 1}
 \end{array}$$

Proof Logging for Advanced Cut Generation: Example

$$C_1 \doteq 6x_1 + 5x_2 + 2x_3 + 3x_4 \geq 8, \quad C_2 \doteq x_1 + x_3 \geq 1$$

- We prove resulting cut $D \doteq 3x_1 + 2x_2 + 2x_3 + 2x_4 \geq 5$ by contradiction
- Can use negation $\neg D \doteq 3x_1 + 2x_2 + 2x_3 + 2x_4 \leq 4 \doteq -3x_1 - 2x_2 - 2x_3 - 2x_4 \geq -4$

$$\begin{array}{r}
 \text{Add} \quad \frac{-3x_1 - 2x_2 - 2x_3 - 2x_4 \geq -4 \quad 6x_1 + 5x_2 + 2x_3 + 3x_4 \geq 8}{3x_1 + 3x_2 + x_4 \geq 4} \\
 \text{Divide by 3} \quad \frac{3x_1 + 3x_2 + x_4 \geq 4}{x_1 + x_2 + x_4 \geq 2} \\
 \text{Multiply by 2} \quad \frac{x_1 + x_2 + x_4 \geq 2}{2x_1 + 2x_2 + 2x_4 \geq 4} \\
 \text{Add} \quad \frac{2x_1 + 2x_2 + 2x_4 \geq 4 \quad -3x_1 - 2x_2 - 2x_3 - 2x_4 \geq -4}{-x_1 - 2x_3 \geq 0} \\
 \text{Add} \quad \frac{-x_1 - 2x_3 \geq 0}{-x_3 \geq 1} \quad \frac{x_1 + x_3 \geq 1}{x_1 + x_3 \geq 1}
 \end{array}$$

pbc +3 x1 +2 x2 +2 x3 +2 x4 >= 5 : subproof

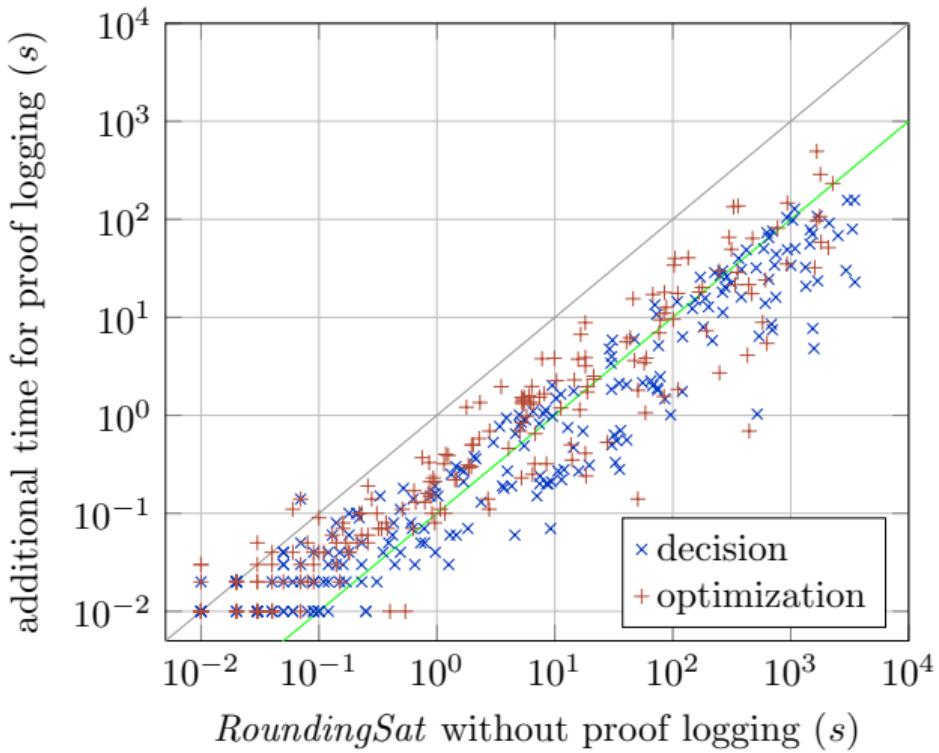
VeriPB: pol -1 @C1 + 3 d 2 * -1 + @C2 + ;
 qed pbc : -1;

Experimental Setup

- Benchmarks: Pseudo-Boolean Competition 2024 [Pse24]
- Run solver without proof logging for 3,600 seconds
- Only consider instances that were solved
- Time limit checker: 36,000 seconds
- Hardware:
 - ▶ i5-1145G7 CPUs
 - ▶ 14GB of available RAM
 - ▶ 100GB solid state drive

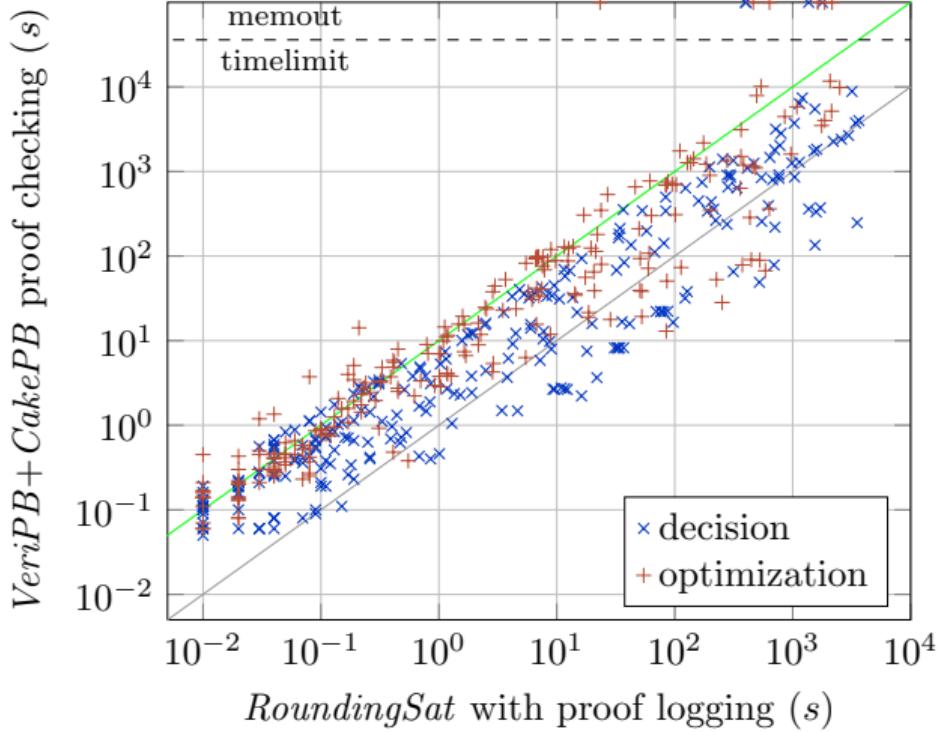
Proof Logging Overhead for *RoundingSat*

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



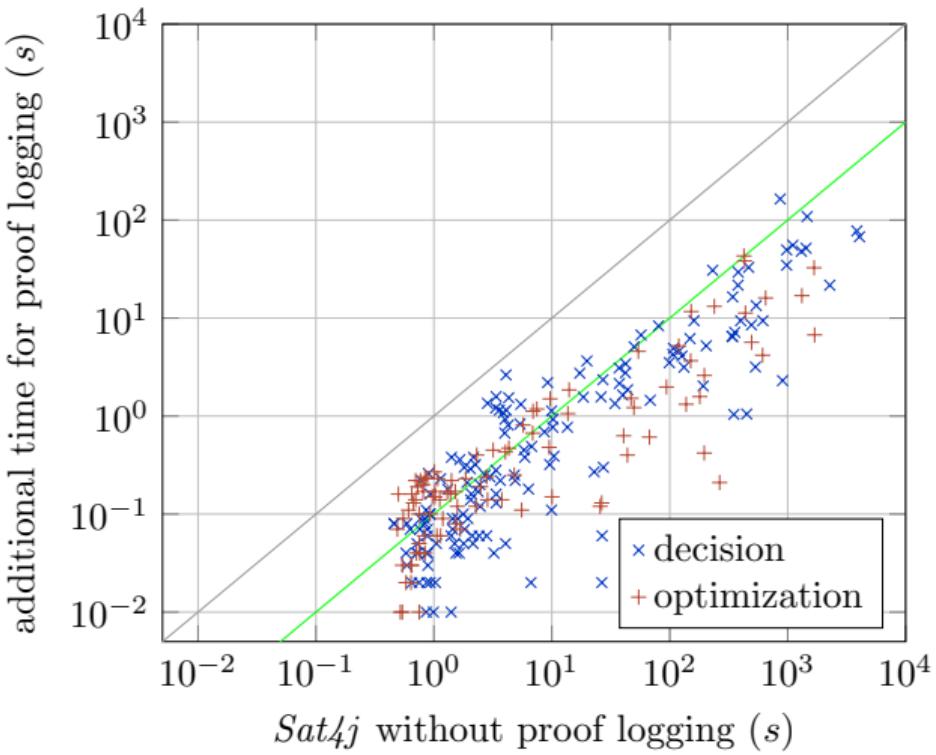
Proof Checking Overhead for *RoundingSat*

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



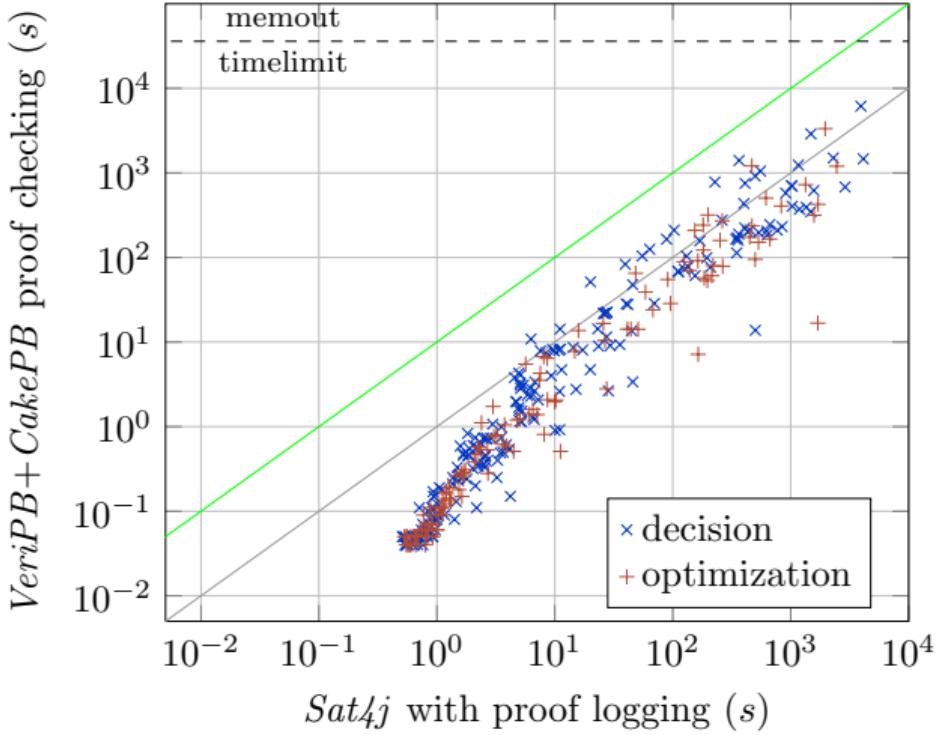
Proof Logging Overhead for *Sat4j*

- Usually $\leq 10\%$
- Worst-case 60%
- Goal: $\leq 10\%$



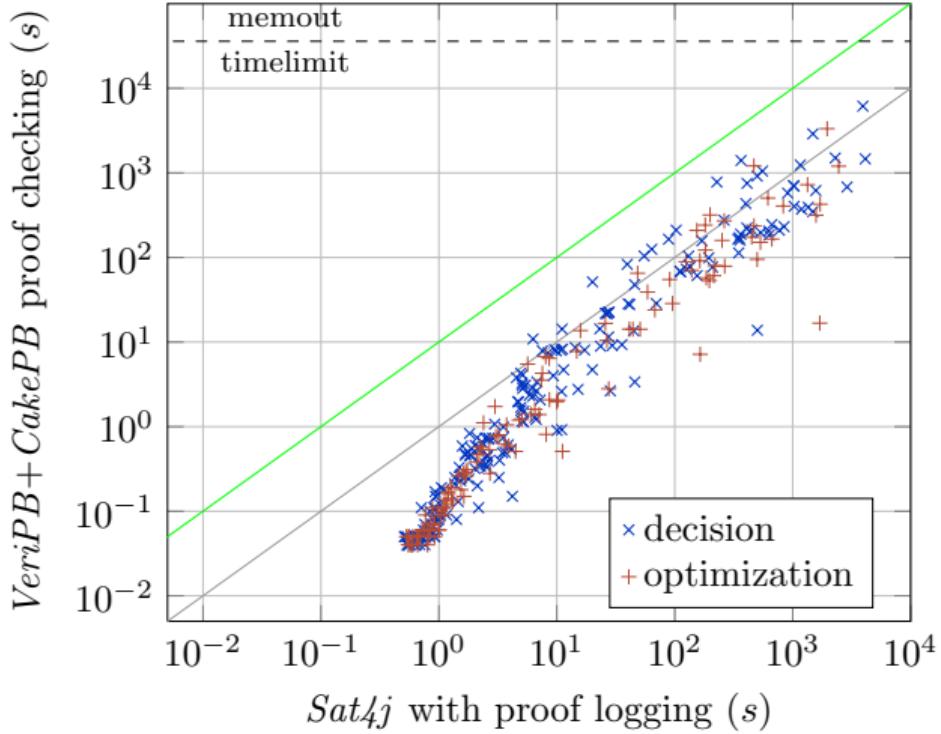
Proof Checking Overhead for *Sat4j*

- Usually $\leq \times 2$
- Worst-case $\times 4$
- Goal: $\leq \times 10$



Proof Checking Overhead for *Sat4j*

- Usually $\leq \times 2$
- Worst-case $\times 4$
- Goal: $\leq \times 10$
- Lower overheads than *RoundingSat*:
 - ▶ Fewer advanced techniques
 - ▶ Java is a bit slower than C++



Using Proof Logging to Detect Inefficiency Bugs

- Main purpose of proof logging: detect **soundness bugs**
- Can also detect bugs leading to **inefficiencies** (but not unsound reasoning)

Using Proof Logging to Detect Inefficiency Bugs

- Main purpose of proof logging: detect **soundness bugs**
- Can also detect bugs leading to **inefficiencies** (but not unsound reasoning)
- Two examples:
 - ▶ Solver used unnecessarily large coefficients compared to constraint derived in proof log
 - ▶ Solver used $Obj \leq v$ instead of objective-improving constraint $Obj \leq v - 1$

Using Proof Logging to Detect Inefficiency Bugs

- Main purpose of proof logging: detect **soundness bugs**
- Can also detect bugs leading to **inefficiencies** (but not unsound reasoning)
- Two examples:
 - ▶ Solver used unnecessarily large coefficients compared to constraint derived in proof log
 - ▶ Solver used $Obj \leq v$ instead of objective-improving constraint $Obj \leq v - 1$
- Having to specify derivation explicitly (in contrast to SAT) can also be an advantage

Challenges for Efficient Proof Logging and Checking

- Attention to detail
 - ▶ Caveat: many low-level details skipped
 - ▶ Getting these right requires in-depth understanding of both solver and *VeriPB*
 - ▶ So efficient proof logging is not just adding a few simple print statements

Challenges for Efficient Proof Logging and Checking

- Attention to detail
 - ▶ Caveat: many low-level details skipped
 - ▶ Getting these right requires in-depth understanding of both solver and *VeriPB*
 - ▶ So efficient proof logging is not just adding a few simple print statements
- Different perspectives in solver and proof checker
 - ▶ *Sat4j* simplifies input constraints but considers them “the same”
 - ▶ In the proof these constraints are clearly different
 - ▶ Requires painful book-keeping during proof logging
 - ▶ New feature of @-labels for constraints very helpful for this

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
 - ▶ ...

Take-Away Message

- This talk:
 - ▶ Survey of some techniques in pseudo-Boolean optimization
 - ▶ Plus explanations how to certify correctness with proof logging
 - ▶ First example of practically feasible certified solving beyond SAT

Take-Away Message

- This talk:
 - ▶ Survey of some techniques in pseudo-Boolean optimization
 - ▶ Plus explanations how to certify correctness with proof logging
 - ▶ 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?

Take-Away Message

- This talk:
 - ▶ Survey of some techniques in pseudo-Boolean optimization
 - ▶ Plus explanations how to certify correctness with proof logging
 - ▶ 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

- [ALB23] Bruno Andreotti, Hanna Lachnitt, and Haniel Barbosa. Carcara: An efficient proof checker and elaborator for SMT proofs in the alethe format. In *Proceedings of the 29th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS '23)*, volume 13993 of *Lecture Notes in Computer Science*, pages 367–386. Springer, April 2023.
- [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

- [BNAH23] Randal E. Bryant, Wojciech Nawrocki, Jeremy Avigad, and Marijn J. H. Heule. Certified knowledge compilation with application to verified model counting. In Meena Mahajan and Friedrich Slivovsky, editors, *26th International Conference on Theory and Applications of Satisfiability Testing, SAT 2023, July 4-8, 2023, Alghero, Italy*, volume 271 of *LIPICS*, pages 6:1–6:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.
- [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.
- [Cap19] Florent Capelli. Knowledge compilation languages as proof systems. In Mikolás Janota and Inês Lynce, editors, *Theory and Applications of Satisfiability Testing - SAT 2019 - 22nd International Conference, SAT 2019, Lisbon, Portugal, July 9-12, 2019, Proceedings*, volume 11628 of *Lecture Notes in Computer Science*, pages 90–99. Springer, 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.

References III

- [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.
- [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*.

References IV

- [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.
- [EN18] Jan Elffers and Jakob Nordström. Divide and conquer: Towards faster pseudo-Boolean solving. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI '18)*, pages 1291–1299, July 2018.
- [FHR22] Johannes Klaus Fichte, Markus Hecher, and Valentin Roland. Proofs for propositional model counting. In Kuldeep S. Meel and Ofer Strichman, editors, *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, August 2-5, 2022, Haifa, Israel*, volume 236 of *LIPICS*, pages 30:1–30:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
- [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.

References V

- [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.
- [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.
- [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

- [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.
- [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.
- [KLM⁺25] Wietze Koops, Daniel Le Berre, Magnus O. Myreen, Jakob Nordström, Andy Oertel, Yong Kiam Tan, and Marc Vinyals. Practically feasible proof logging for pseudo-Boolean optimization. In *Proceedings of the 31st International Conference on Principles and Practice of Constraint Programming (CP '25)*, August 2025. To appear.

References VII

- [LP10] Daniel Le Berre and Anne Parrain. The Sat4j library, release 2.2. *Journal on Satisfiability, Boolean Modeling and Computation*, 7:59–64, July 2010.
- [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.
- [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.

References VIII

- [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.
- [Pse24] Pseudo-Boolean competition 2024. <https://www.cril.univ-artois.fr/PB24/>, August 2024.
- [SFBF21] Hans-Jörg Schurr, Mathias Fleury, Haniel Barbosa, and Pascal Fontaine. Alethe: Towards a generic SMT proof format (extended abstract). In *Proceedings of the 7th Workshop on Proof eXchange for Theorem Proving (PxTP '21)*, volume 336 of *Electronic Proceedings in Theoretical Computer Science*, pages 49–54, July 2021.
- [VS10] Michael Veksler and Ofer Strichman. A proof-producing CSP solver. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI '10)*, pages 204–209, July 2010.
- [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.