Certified CNF Translations for Pseudo-Boolean Solving

Jakob Nordström

University of Copenhagen
and Lund University

Swedish Operations Research Conference (SOAK 2022)

October 24, 2022

Joint work with Stephan Gocht, Ruben Martins, and Andy Oertel

Certified CNF Translations for Pseudo-Boolean Solving

Jakob Nordström

University of Copenhagen
and Lund University

Swedish Operations Research Conference (SOAK 2022)

October 24, 2022

Joint work with Stephan Gocht, Ruben Martins, and Andy Oertel

Thanks for the slides!

The Pseudo-Boolean (PB) Problem

pseudo-Boolean formula:

$$x_1 + x_2 + x_3 \geq 2$$
$$x_2 + x_3 + x_4 \geq 2$$
$$\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 + \overline{x}_4 \geq 3$$

$\rightarrow$ Pseudo-Boolean Solver $\rightarrow$

Result: SAT/UNSAT

- Input: Pseudo-Boolean formula (a.k.a. 0-1 integer linear program)
  - Collection of 0-1 integer linear constraints

The Pseudo-Boolean (PB) Problem

0-1 integer linear constraints:

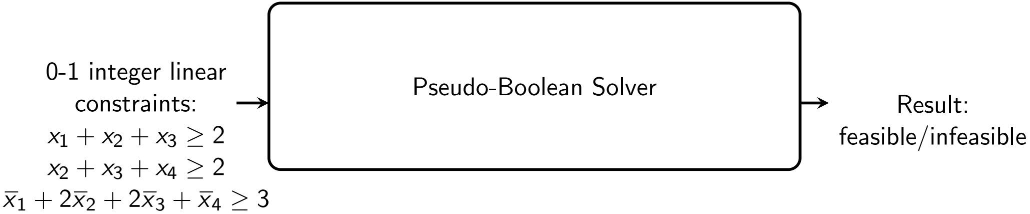
$$x_1 + x_2 + x_3 \geq 2$$
$$x_2 + x_3 + x_4 \geq 2$$
$$\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 + \overline{x}_4 \geq 3$$

Pseudo-Boolean Solver

Result: feasible/infeasible

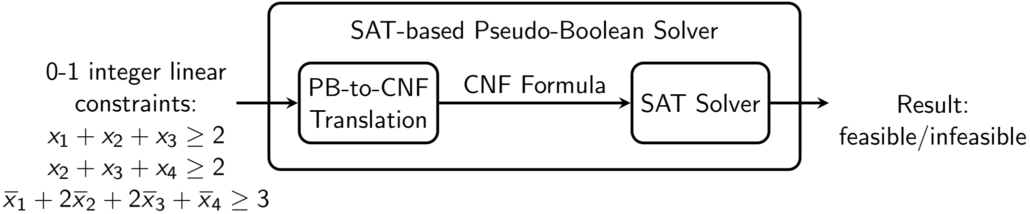
- Input: Pseudo-Boolean formula (a.k.a. 0-1 integer linear program)
  - Collection of 0-1 integer linear constraints

The Pseudo-Boolean (PB) Problem



- Input: Pseudo-Boolean formula (a.k.a. 0-1 integer linear program)
  - Collection of 0-1 integer linear constraints
- Pseudo-Boolean solvers:
  - Native: Sat4j [LP10], RoundingSAT [EN18]
  - SAT-based: MiniSAT+ [ES06], Open-WBO [MML14], NaPS [SN15]

The Pseudo-Boolean (PB) Problem



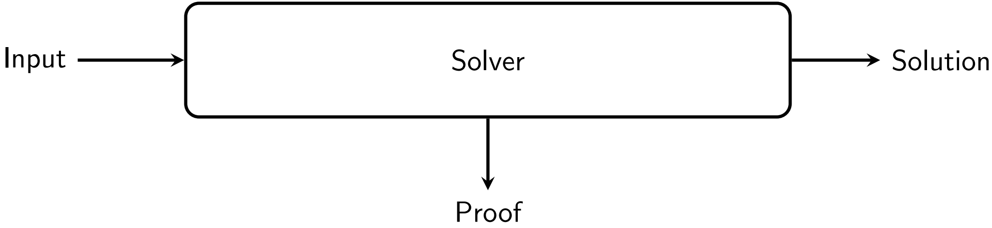
- **Input:** Pseudo-Boolean formula (a.k.a. 0-1 integer linear program)
  - Collection of 0-1 integer linear constraints
- **Pseudo-Boolean solvers:**
  - Native: Sat4j [LP10], RoundingSAT [EN18]
  - **SAT-based:** MiniSAT+ [ES06], Open-WBO [MML14], NaPS [SN15]

Certification with Proof Logging

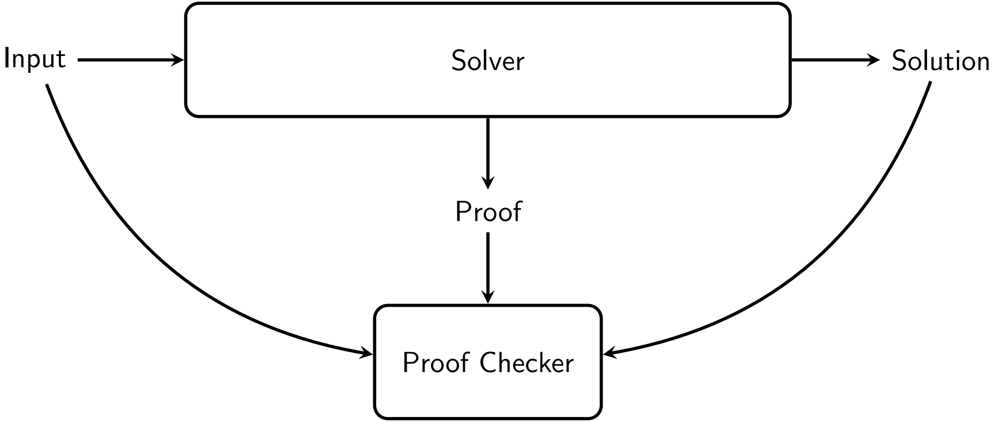
Solver

Certification with Proof Logging

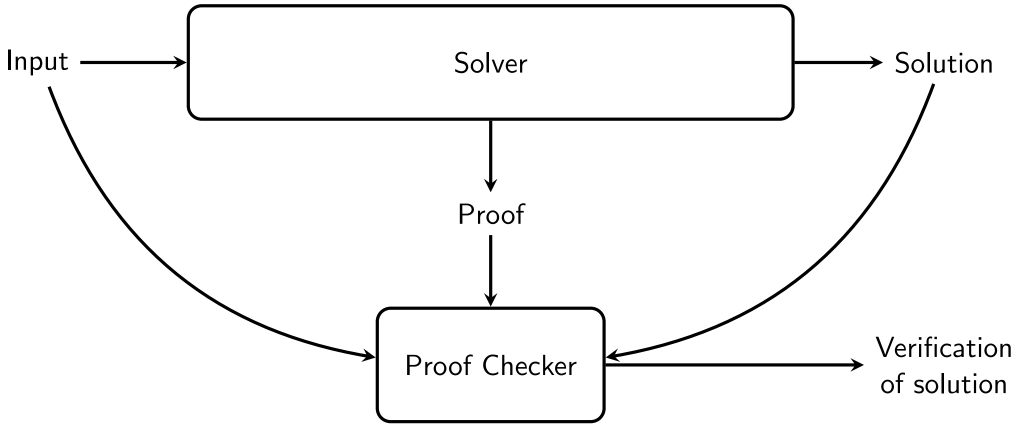
Certification with Proof Logging



Certification with Proof Logging



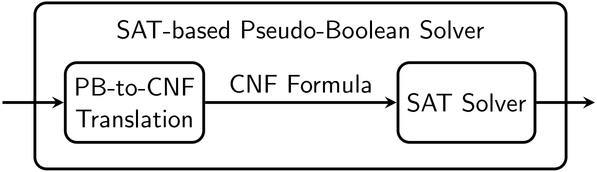
Certification with Proof Logging



Certifying Correctness

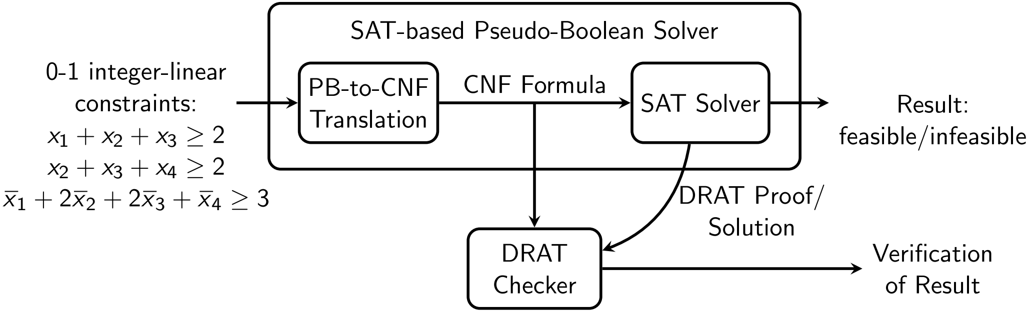
0-1 integer-linear constraints:

$$x_1 + x_2 + x_3 \geq 2$$
$$x_2 + x_3 + x_4 \geq 2$$
$$\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 + \overline{x}_4 \geq 3$$



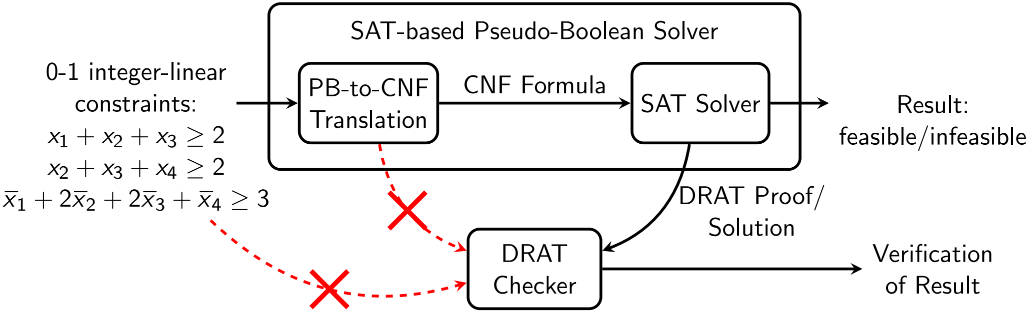
Result: feasible/infeasible

Certifying Correctness



- Correctness of SAT solver solution can be certified [HHW13a, HHW13b, WHH14]

Certifying Correctness



- Correctness of SAT solver solution can be certified [HHW13a, HHW13b, WHH14]
- **PB-to-CNF translation uncertified!**

Pseudo-Boolean Proof Logging

- **Multi-purpose** proof format
- Allows easy proof logging for
  - Reasoning with pseudo-Boolean constraints (by design)
  - SAT solving (including advanced techniques) [GN21, BGMN22]
  - Constraint programming [EGMN20, GMN22]
  - Subgraph problems [GMN20, GMM+20]

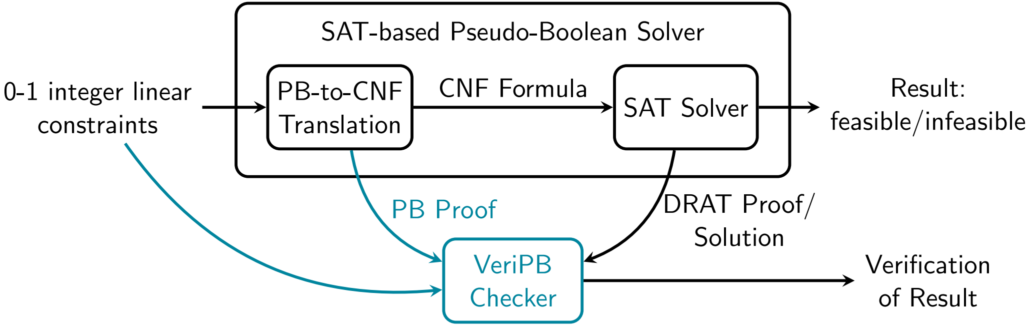
Pseudo-Boolean Proof Logging

- **Multi-purpose** proof format
- Allows easy proof logging for
  - Reasoning with pseudo-Boolean constraints (by design)
  - SAT solving (including advanced techniques) [GN21, BGMN22]
  - Constraint programming [EGMN20, GMN22]
  - Subgraph problems [GMN20, GMM+20]

This Work

- Proof logging for translating pseudo-Boolean constraints to CNF
- **General framework** to certify many different encodings
- Promising foundation for certifying MaxSAT solving and PB optimization

Workflow



SAT-based Pseudo-Boolean Solver
0-1 integer linear constraints
PB-to-CNF Translation
CNF Formula
SAT Solver
Result: feasible/infeasible
PB Proof
DRAT Proof/ Solution
VeriPB Checker
Verification of Result

Basic Notation

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

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

Basic Notation

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

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

- Equality constraint: syntactic sugar for 2 inequalities

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

Basic Notation

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

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

- Equality constraint: syntactic sugar for 2 inequalities

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

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

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

Cutting Planes Proof System [CCT87]

Rules:

- Literal axiom

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

Cutting Planes Proof System [CCT87]

Rules:

- Literal axiom

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

- Addition

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

Cutting Planes Proof System [CCT87]

Rules:

- Literal axiom

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

- Addition

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

- Multiplication

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

Cutting Planes Proof System [CCT87]

Rules:

- Literal axiom

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

- Addition

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

- Multiplication

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

- Division

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

Extended Cutting Planes: Reification

Extension rule to introduce fresh variables:

- Reification (special case of redundance rule in [GN21, BGMN22])

$$a \Leftrightarrow x_1 + \overline{x}_2 + 2x_3 \geq 2 \longrightarrow \begin{array}{ll} 2\overline{a} + x_1 + \overline{x}_2 + 2x_3 \geq 2 & \qquad (a \Rightarrow x_1 + \overline{x}_2 + 2x_3 \geq 2) \\ 3a + \overline{x}_1 + x_2 + 2\overline{x}_3 \geq 3 & \qquad (a \Leftarrow x_1 + \overline{x}_2 + 2x_3 \geq 2) \end{array}$$

Translating 0-1 ILP to CNF: Outline

1. Construct circuit evaluating left-hand side of 0-1 integer linear constraint
2. Encode circuit to CNF using so-called Tseitin translation
3. Enforce constraint

Translating 0-1 ILP to CNF: Step 1

1. **Construct circuit evaluating left-hand side of 0-1 integer linear constraint**

- Several approaches to construct logical circuit evaluating PB constraint
  - Sequential counter [Sin05], totalizer [BB03], adder network [ES06], ...

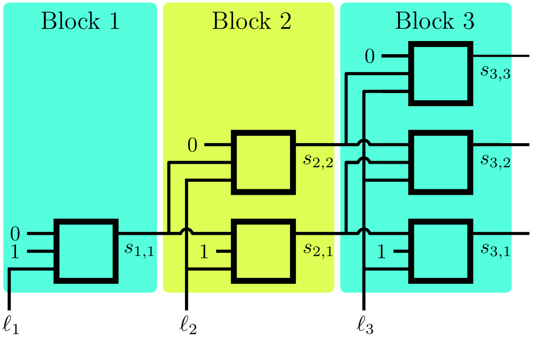
Translating 0-1 ILP to CNF: Step 1

1. **Construct circuit evaluating left-hand side of 0-1 integer linear constraint**

- Several approaches to construct logical circuit evaluating PB constraint
  - **Sequential counter** [Sin05], totalizer [BB03], adder network [ES06], ...

Example: $\ell_1 + \ell_2 + \ell_3 \geq 2$

Meaning of $s_{i,j}$ variable:
$s_{i,j}$ true if and only if
$\ell_1 + \ldots + \ell_i \geq j$



Translating 0-1 ILP to CNF: Step 2

2. **Encode circuit to CNF using so-called Tseitin translations**

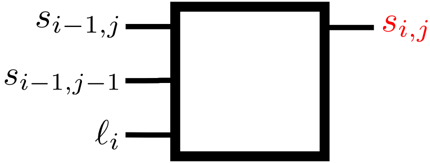
- Introduce fresh variable for each wire
- Encode using clauses describing behavior of each component

Translating 0-1 ILP to CNF: Step 2

2. **Encode circuit to CNF using so-called Tseitin translations**

- Introduce fresh variable for each wire
- Encode using clauses describing behavior of each component

Example: Sequential counter component



Specification of $s_{i,j}$

$$s_{i,j} \leftrightarrow (\ell_i \wedge s_{i-1,j-1}) \vee s_{i-1,j}$$

Clausal encoding

$$\overline{\ell}_i \vee \overline{s}_{i-1,j-1} \vee s_{i,j}$$
$$\overline{s}_{i-1,j} \vee s_{i,j}$$
$$\ell_i \vee s_{i-1,j} \vee \overline{s}_{i,j}$$
$$s_{i-1,j-1} \vee \overline{s}_{i,j}$$

Translating 0-1 ILP to CNF: Step 3

3. **Enforce constraint**

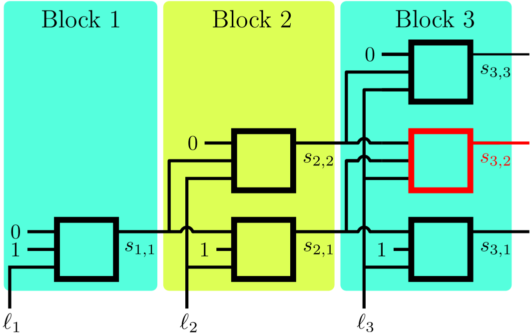
- Add clauses enforcing comparison with right-hand side

Translating 0-1 ILP to CNF: Step 3

3. **Enforce constraint**

- Add clauses enforcing comparison with right-hand side

Example: $\ell_1 + \ell_2 + \ell_3 \geq 2$



At least 2 true literals if $s_{3,2}$ true.

Add unary clause

$s_{3,2}$

Our Work: Translation Correct?

Is the translation correct?

Our Work: Translation Correct?

Is the translation correct?

- **Yes!** Sinz showed that in [Sin05]
- But did we code up the translation correctly?

Our Work: Translation Correct?

Is the translation correct?

- **Yes!** Sinz showed that in [Sin05]
- But did we code up the translation correctly?

How can we show correctness?

- Proof logging!
- In our case: Give formal derivation of clauses using Cutting Planes + reification

Our Work: Translation Correct?

Is the translation correct?

- **Yes!** Sinz showed that in [Sin05]
- But did we code up the translation correctly?

How can we show correctness?

- Proof logging!
- In our case: Give formal derivation of clauses using Cutting Planes + reification

This means

- 0-1 ILP has feasible solutions $\Longrightarrow$ CNF translation satisfiable
- Solver finds no solution to CNF translation $\Longrightarrow$ 0-1 ILP is infeasible

Our Work: Translation Correct?

Is the translation correct?

- **Yes!** Sinz showed that in [Sin05]
- But did we code up the translation correctly?

How can we show correctness?

- Proof logging!
- In our case: Give formal derivation of clauses using Cutting Planes + reification

This means

- 0-1 ILP has feasible solutions $\Longrightarrow$ CNF translation satisfiable
- Solver finds no solution to CNF translation $\Longrightarrow$ 0-1 ILP is infeasible

End-to-end verification of SAT-based pseudo-Boolean solvers!

Rest of This Talk: Some technical details?

We develop general framework certifying PB-to-CNF translations

- But let us stay with our example:

Sequential counter encoding of $\ell_1 + \ell_2 + \ell_3 \geq 2$

Circuit Specification in Pseudo-Boolean Form

Using Cutting Planes + reification, do syntactic derivation of circuit specification:

- Specification of $s_{i,j}$ variables

$$s_{i,j} \Leftrightarrow \sum_{k=1}^{i-1} s_{i-1,k} + \ell_i \geq j$$

Circuit Specification in Pseudo-Boolean Form

Using Cutting Planes + reification, do syntactic derivation of circuit specification:

- Specification of $s_{i,j}$ variables

$$s_{i,j} \Leftrightarrow \sum_{k=1}^{i-1} s_{i-1,k} + \ell_i \geq j$$

- Ordering of $s_{i,j}$ variables

$$s_{i,j} \geq s_{i,j+1}$$

Circuit Specification in Pseudo-Boolean Form

Using Cutting Planes + reification, do syntactic derivation of circuit specification:

- Specification of $s_{i,j}$ variables

$$s_{i,j} \Leftrightarrow \sum_{k=1}^{i-1} s_{i-1,k} + \ell_i \geq j$$

- Ordering of $s_{i,j}$ variables

$$s_{i,j} \geq s_{i,j+1}$$

- Preservation of sum

$$\sum_{k=1}^{i} s_{i,k} = \sum_{k=1}^{i-1} s_{i-1,k} + \ell_i$$

Deriving the CNF Translation

We now have 0-1 integer linear constraints:

$$s_{1,1} = \ell_1 \qquad s_{2,1} + s_{2,2} = s_{1,1} + \ell_2 \qquad s_{3,1} + s_{3,2} + s_{3,3} = s_{2,1} + s_{2,2} + \ell_3$$
$$s_{2,1} \geq s_{2,2} \qquad s_{3,1} \geq s_{3,2} \qquad s_{3,2} \geq s_{3,3} \qquad s_{3,1} + s_{3,2} + s_{3,3} \geq 2$$

Deriving the CNF Translation

We now have 0-1 integer linear constraints:

$$s_{1,1} = \ell_1 \qquad s_{2,1} + s_{2,2} = s_{1,1} + \ell_2 \qquad s_{3,1} + s_{3,2} + s_{3,3} = s_{2,1} + s_{2,2} + \ell_3$$

$$s_{2,1} \geq s_{2,2} \qquad s_{3,1} \geq s_{3,2} \qquad s_{3,2} \geq s_{3,3} \qquad s_{3,1} + s_{3,2} + s_{3,3} \geq 2$$

But we want clauses:

$$\begin{array}{rrrr}
\overline{\ell}_1 \lor s_{1,1} & \overline{\ell}_2 \lor \overline{s}_{1,1} \lor s_{2,2} & \ell_3 \lor s_{2,1} \lor \overline{s}_{3,1} & \overline{\ell}_3 \lor \overline{s}_{2,2} \lor s_{3,3} \\
\ell_1 \lor \overline{s}_{1,1} & \ell_2 \lor \overline{s}_{2,2} & \overline{\ell}_3 \lor \overline{s}_{2,1} \lor s_{3,2} & \ell_3 \lor \overline{s}_{3,3} \\
\overline{\ell}_2 \lor s_{2,1} & s_{1,1} \lor \overline{s}_{2,2} & \overline{s}_{2,2} \lor s_{3,2} & s_{2,2} \lor \overline{s}_{3,3} \\
\overline{s}_{1,1} \lor s_{2,1} & \overline{\ell}_3 \lor s_{3,1} & \ell_3 \lor s_{2,2} \lor \overline{s}_{3,2} & s_{3,2} \\
\ell_2 \lor s_{1,1} \lor \overline{s}_{2,1} & \overline{s}_{2,1} \lor s_{3,1} & s_{2,1} \lor \overline{s}_{3,2} &
\end{array}$$

Deriving the CNF Translation

We now have 0-1 integer linear constraints:

$$s_{1,1} = \ell_1 \qquad s_{2,1} + s_{2,2} = s_{1,1} + \ell_2 \qquad s_{3,1} + s_{3,2} + s_{3,3} = s_{2,1} + s_{2,2} + \ell_3$$
$$s_{2,1} \geq s_{2,2} \qquad s_{3,1} \geq s_{3,2} \qquad s_{3,2} \geq s_{3,3} \qquad s_{3,1} + s_{3,2} + s_{3,3} \geq 2$$

But we want clauses:

$$\begin{array}{rrrr}
\bar{\ell}_1 \vee s_{1,1} & \bar{\ell}_2 \vee \bar{s}_{1,1} \vee s_{2,2} & \ell_3 \vee s_{2,1} \vee \bar{s}_{3,1} & \bar{\ell}_3 \vee \bar{s}_{2,2} \vee s_{3,3} \\
\ell_1 \vee \bar{s}_{1,1} & \ell_2 \vee \bar{s}_{2,2} & \bar{\ell}_3 \vee \bar{s}_{2,1} \vee s_{3,2} & \ell_3 \vee \bar{s}_{3,3} \\
\bar{\ell}_2 \vee s_{2,1} & s_{1,1} \vee \bar{s}_{2,2} & \bar{s}_{2,2} \vee s_{3,2} & s_{2,2} \vee \bar{s}_{3,3} \\
\bar{s}_{1,1} \vee s_{2,1} & \bar{\ell}_3 \vee s_{3,1} & \ell_3 \vee s_{2,2} \vee \bar{s}_{3,2} & s_{3,2} \\
\ell_2 \vee s_{1,1} \vee \bar{s}_{2,1} & \bar{s}_{2,1} \vee s_{3,1} & s_{2,1} \vee \bar{s}_{3,2} &
\end{array}$$

- Follow easily from PB specification by so-called reverse unit propagation [GN03, Van08]
- See SAT'22 paper for details [GMNO22]

Experiments

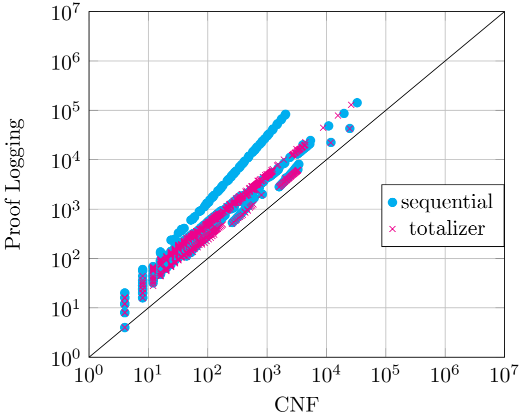
- Certified translations for the following CNF encodings:[1]
  - Sequential counter [Sin05]
  - Totalizer [BB03]
  - Generalized totalizer [JMM15]
  - Adder network [ES06]

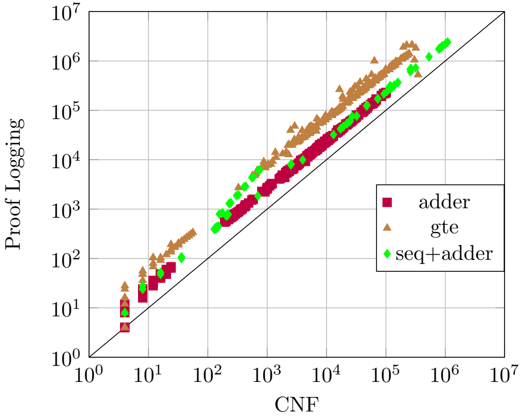
- Proof verified by proof checker VeriPB[2]

- Benchmarks from PB 2016 Evaluation:[3]
  - `SMALLINT` decision benchmarks without purely clausal formulas
  - 3 subclasses of benchmarks:
    - Only cardinality constraints (sequential counter, totalizer)
    - Only general 0-1 ILP constraints (generalized totalizer, adder network)
    - Mixed cardinality & general 0-1 ILP constraints (sequential counter + adder network)

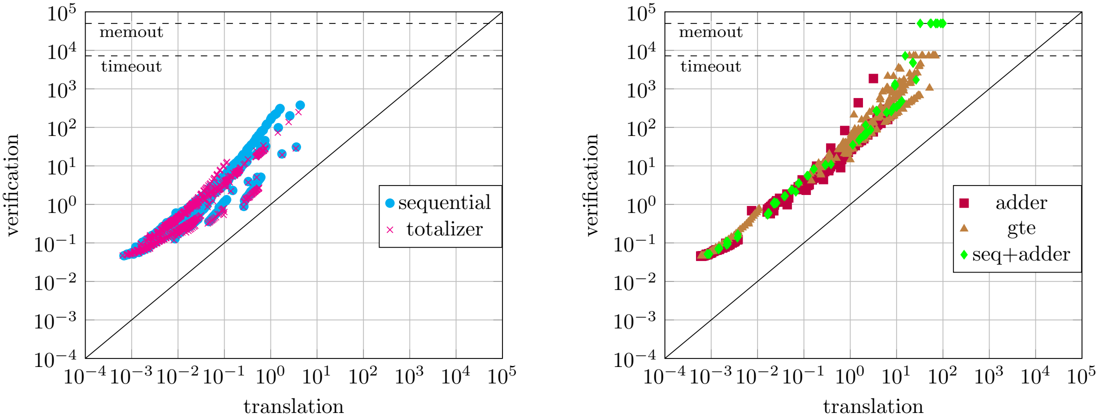
[1] `https://github.com/forge-lab/VeritasPBLib`
[2] `https://gitlab.com/MIAOresearch/software/VeriPB`
[3] `http://www.cril.univ-artois.fr/PB16/`

CNF Size vs Proof Size in KiB



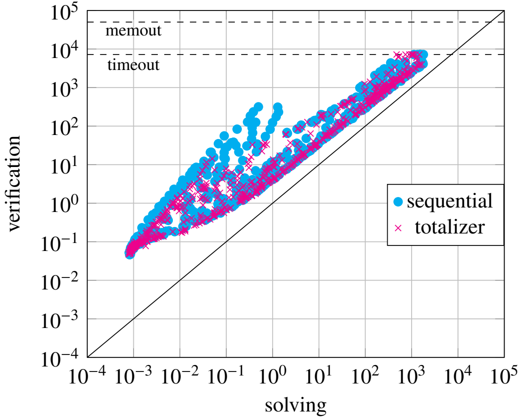


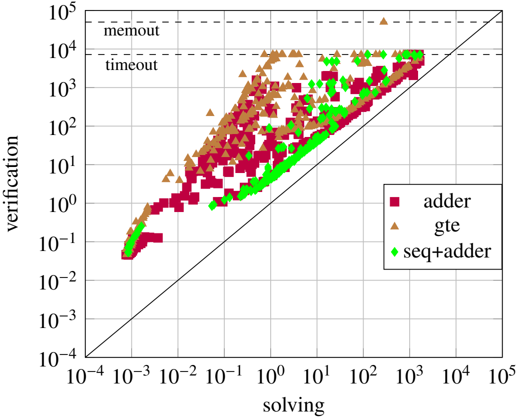
Translation Time vs Verification Time in Seconds



- Translation just generates clauses and proof
- Verification slower, as reasoning has to be performed

Solving Time vs Verification Time in Seconds





- Solved with fork of Kissat[4] syntactically modified to output pseudo-Boolean proofs
- Room for improvement, but this clearly shows that our approach is viable

[4] `https://gitlab.com/MIAOresearch/tools-and-utilities/kissat_fork`

Future Work

Improving performance:

- Cutting Planes derivations instead of reverse unit propagations [VDB22]
- Backwards checking/trimming for verification (as in DRAT-trim [HHW13a])

Future Work

Improving performance:

- Cutting Planes derivations instead of reverse unit propagations [VDB22]
- Backwards checking/trimming for verification (as in DRAT-trim [HHW13a])

Extend proof logging further:

- Sorting networks like odd-even mergesort, bitonic sorter [Bat68]
- MaxSAT solving and pseudo-Boolean optimization
- Mixed integer linear programming

Conclusion

This work:

- General approach for certifying different PB-to-CNF translations
- End-to-end verification of SAT-based pseudo-Boolean solving

Conclusion

This work:

- General approach for certifying different PB-to-CNF translations
- End-to-end verification of SAT-based pseudo-Boolean solving

Pseudo-Boolean reasoning provides unified proof logging method for:

- SAT solving (including advanced techniques) [GN21, BGMN22]
- (Basic) constraint programming [EGMN20, GMN22]
- Subgraph problems [GMN20, GMM$^+$20]
- **This work:** SAT-based pseudo-Boolean solving

Conclusion

This work:

- General approach for certifying different PB-to-CNF translations
- End-to-end verification of SAT-based pseudo-Boolean solving

Pseudo-Boolean reasoning provides unified proof logging method for:

- SAT solving (including advanced techniques) [GN21, BGMN22]
- (Basic) constraint programming [EGMN20, GMN22]
- Subgraph problems [GMN20, GMM+20]
- **This work:** SAT-based pseudo-Boolean solving
- **Next up:** MaxSAT solving and pseudo-Boolean optimization

Conclusion

This work:

- General approach for certifying different PB-to-CNF translations
- End-to-end verification of SAT-based pseudo-Boolean solving

Pseudo-Boolean reasoning provides unified proof logging method for:

- SAT solving (including advanced techniques) [GN21, BGMN22]
- (Basic) constraint programming [EGMN20, GMN22]
- Subgraph problems [GMN20, GMM+20]
- **This work:** SAT-based pseudo-Boolean solving
- **Next up:** MaxSAT solving and pseudo-Boolean optimization
- **Future goal(?):** Mixed integer linear programming

Conclusion

This work:

- General approach for certifying different PB-to-CNF translations
- End-to-end verification of SAT-based pseudo-Boolean solving

Pseudo-Boolean reasoning provides unified proof logging method for:

- SAT solving (including advanced techniques) [GN21, BGMN22]
- (Basic) constraint programming [EGMN20, GMN22]
- Subgraph problems [GMN20, GMM+20]
- **This work:** SAT-based pseudo-Boolean solving
- **Next up:** MaxSAT solving and pseudo-Boolean optimization
- **Future goal(?):** Mixed integer linear programming

Thank you for your attention!

References I

[Bat68] Kenneth E. Batcher.
Sorting networks and their applications.
In *Proceedings of the Spring Joint Computer Conference of the American Federation of Information Processing Societies (AFIPS '68)*, volume 32, pages 307–314, April 1968.

[BB03] Olivier Bailleux and Yacine Boufkhad.
Efficient CNF encoding of Boolean cardinality constraints.
In *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming (CP '03)*, volume 2833 of *Lecture Notes in Computer Science*, pages 108–122. Springer, September 2003.

[BGMN22] Bart Bogaerts, Stephan Gocht, Ciaran McCreesh, and Jakob Nordström.
Certified symmetry and dominance breaking for combinatorial optimisation.
In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI '22)*, pages 3698–3707, February 2022.

[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 II

[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.

[ES06] Niklas Eén and Niklas Sörensson.
Translating pseudo-Boolean constraints into SAT.
Journal on Satisfiability, Boolean Modeling and Computation, 2(1-4):1–26, March 2006.

[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 III

[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.

[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.

[GMNO22] Stephan Gocht, Ruben Martins, Jakob Nordström, and Andy Oertel.
Certified CNF translations for pseudo-Boolean solving.
In *Proceedings of the 25th International Conference on Theory and Applications of Satisfiability Testing (SAT '22)*, volume 236 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 16:1–16:25, August 2022.

[GN03] Evgueni Goldberg and Yakov Novikov.
Verification of proofs of unsatisfiability for CNF formulas.
In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE '03)*, pages 886–891, March 2003.

References IV

[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.

[HHW13a] 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.

[HHW13b] Marijn J. H. Heule, Warren A. Hunt Jr., and Nathan Wetzler.
Verifying refutations with extended resolution.
In *Proceedings of the 24th International Conference on Automated Deduction (CADE-24)*, volume 7898 of *Lecture Notes in Computer Science*, pages 345–359. Springer, June 2013.

[JMM15] Saurabh Joshi, Ruben Martins, and Vasco M. Manquinho.
Generalized totalizer encoding for pseudo-Boolean constraints.
In *Proceedings of the 21st International Conference on Principles and Practice of Constraint Programming (CP '15)*, volume 9255 of *Lecture Notes in Computer Science*, pages 200–209. Springer, August-September 2015.

References V

[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.

[MML14] Ruben Martins, Vasco M. Manquinho, and Inês Lynce.
Open-WBO: A modular MaxSAT solver.
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 438–445. Springer, July 2014.

[Sin05] Carsten Sinz.
Towards an optimal CNF encoding of Boolean cardinality constraints.
In *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming (CP '05)*, volume 3709 of *Lecture Notes in Computer Science*, pages 827–831. Springer, October 2005.

[SN15] Masahiko Sakai and Hidetomo Nabeshima.
Construction of an ROBDD for a PB-constraint in band form and related techniques for PB-solvers.
IEICE Transactions on Information and Systems, 98-D(6):1121–1127, June 2015.

References VI

[Van08] Allen Van Gelder.
Verifying RUP proofs of propositional unsatisfiability.
In *10th International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, 2008.
http://isaim2008.unl.edu/index.php?page=proceedings.

[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.