Certified CNF Translations for Pseudo-Boolean Solving

Jakob Nordström

University of Copenhagen
and Lund University

32nd International Joint Conference
on Artificial Intelligence
Macau, China
August 19–25, 2023

SAT '22 paper joint with Stephan Gocht, Ruben Martins, and Andy Oertel

Pseudo-Boolean (PB) Solving

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

Pseudo-Boolean (PB) Solving

0–1 integer linear program:

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

$\rightarrow$ Pseudo-Boolean solver $\rightarrow$

Result: feasible/infeasible

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

Pseudo-Boolean (PB) Solving

0–1 integer linear program:

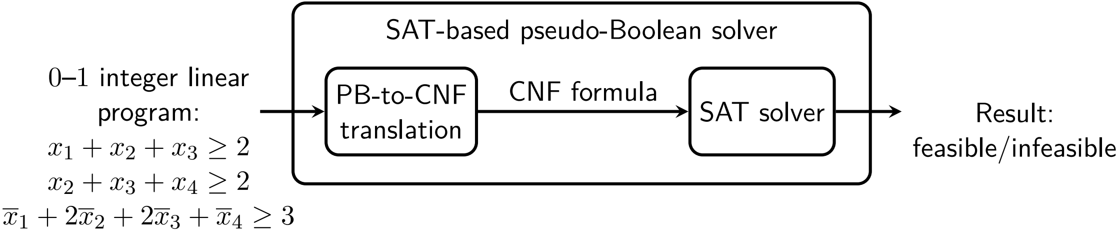
$$x_1 + x_2 + x_3 \geq 2$$
$$x_2 + x_3 + x_4 \geq 2$$
$$\overline{x}_1 + 2\overline{x}_2 + 2\overline{x}_3 + \overline{x}_4 \geq 3$$

→ 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
- Conflict-driven pseudo-Boolean solvers:
  - Native: SAT4J [LP10], ROUNDINGSAT [EN18]
  - SAT-based: MINISAT+ [ES06], OPEN-WBO [MML14], NAPS [SN15]

Pseudo-Boolean (PB) Solving

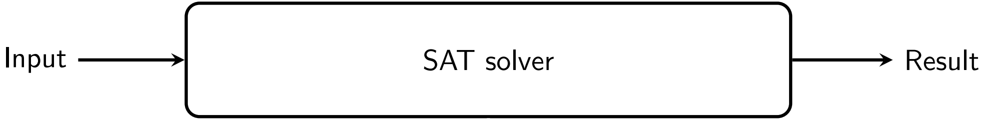


- Input: Pseudo-Boolean formula (a.k.a. 0–1 integer linear program)
  - Collection of 0–1 integer linear constraints
- Conflict-driven pseudo-Boolean solvers:
  - Native: SAT4J [LP10], ROUNDINGSAT [EN18]
  - **SAT-based:** MINISAT+ [ES06], OPEN-WBO [MML14], NAPS [SN15]

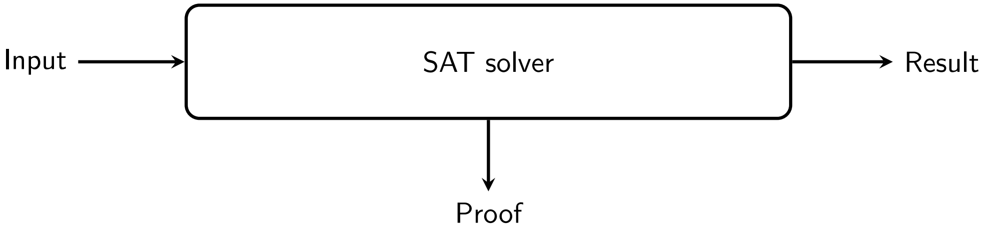
Certifying SAT Solver Results with Proof Logging

SAT solver

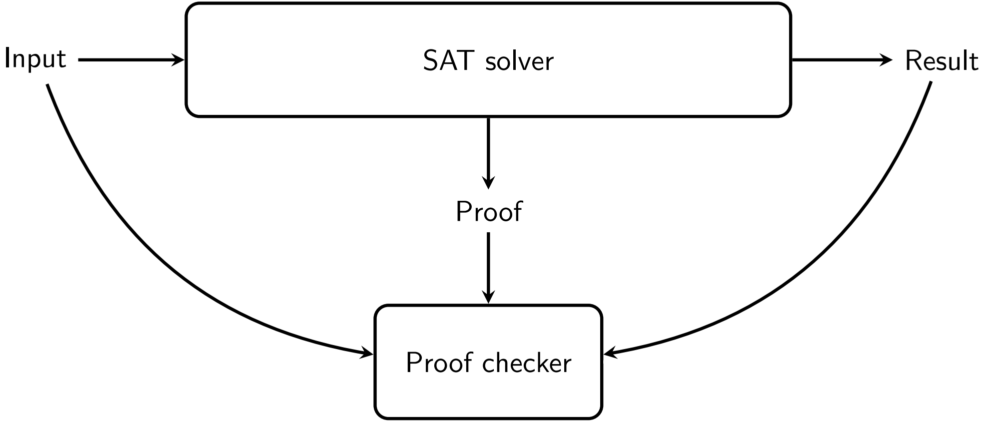
Certifying SAT Solver Results with Proof Logging



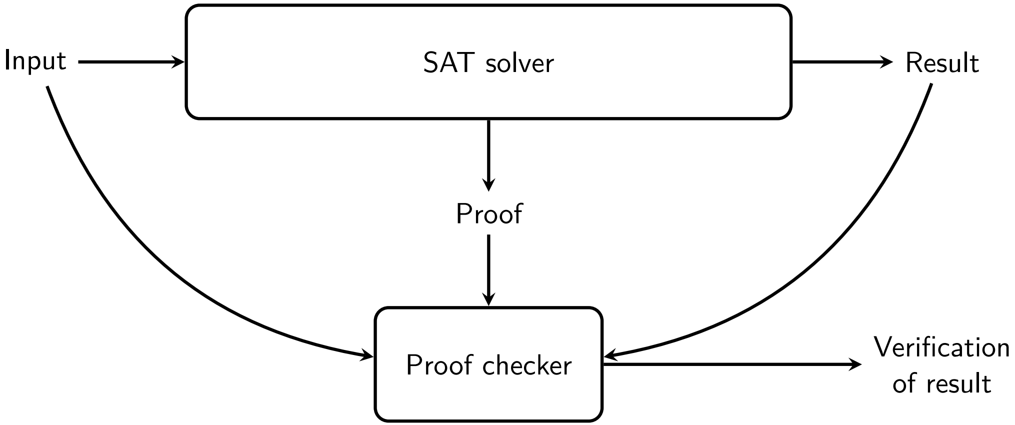
Certifying SAT Solver Results with Proof Logging



Certifying SAT Solver Results with Proof Logging



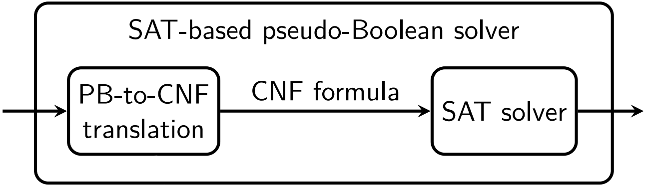
Certifying SAT Solver Results with Proof Logging



Certifying Pseudo-Boolean Solver Results with Proof Logging?

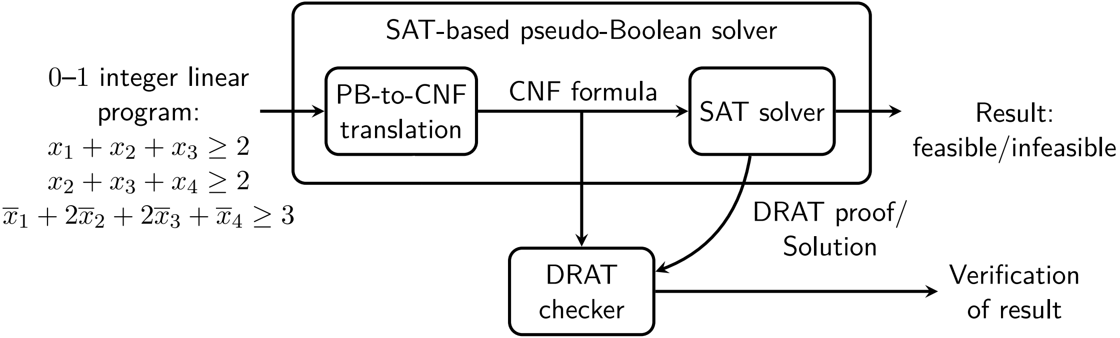
0–1 integer linear program:

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



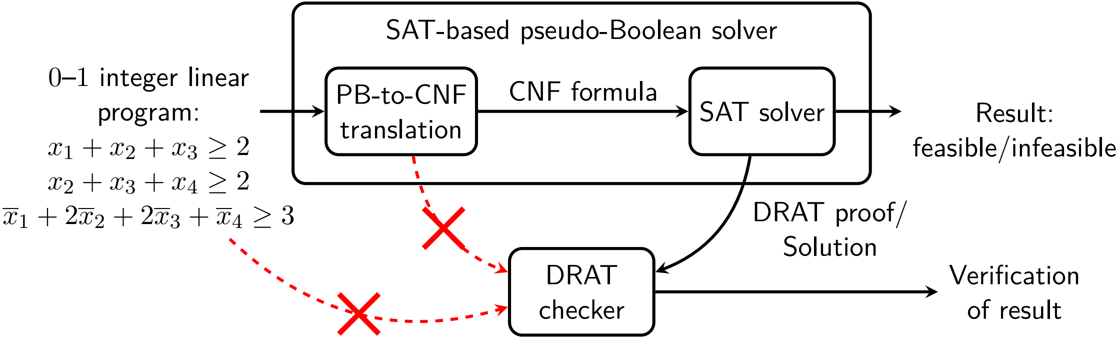
Result: feasible/infeasible

Certifying Pseudo-Boolean Solver Results with Proof Logging?



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

Certifying Pseudo-Boolean Solver Results with Proof Logging?



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

Example Pseudo-Boolean-to-CNF Translation

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

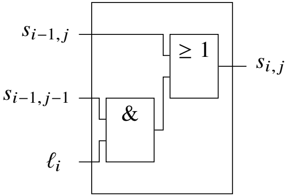
Example Pseudo-Boolean-to-CNF Translation

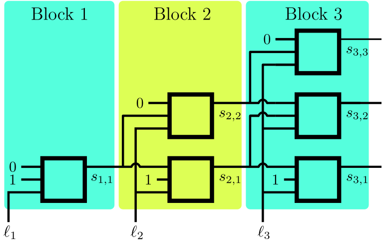
$\ell_1 + \ell_2 + \ell_3 \geq 2$

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

Example Pseudo-Boolean-to-CNF Translation

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





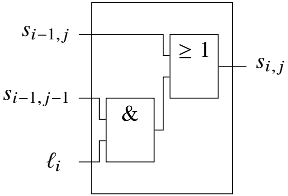
$$
\begin{array}{rrrr}
\bar\ell_1 \vee s_{1,1} & \bar\ell_2 \vee \bar s_{1,1} \vee s_{2,2} & \ell_3 \vee s_{2,1} \vee \bar s_{3,1} & \bar\ell_3 \vee \bar s_{2,2} \vee s_{3,3} \\
\ell_1 \vee \bar s_{1,1} & \ell_2 \vee \bar s_{2,2} & \bar\ell_3 \vee \bar s_{2,1} \vee s_{3,2} & \ell_3 \vee \bar s_{3,3} \\
\bar\ell_2 \vee s_{2,1} & s_{1,1} \vee \bar s_{2,2} & \bar s_{2,2} \vee s_{3,2} & s_{2,2} \vee \bar s_{3,3} \\
\bar s_{1,1} \vee s_{2,1} & \bar\ell_3 \vee s_{3,1} & \ell_3 \vee s_{2,2} \vee \bar s_{3,2} & s_{3,2} \\
\ell_2 \vee s_{1,1} \vee \bar s_{2,1} & \bar s_{2,1} \vee s_{3,1} & s_{2,1} \vee \bar s_{3,2} &
\end{array}
$$

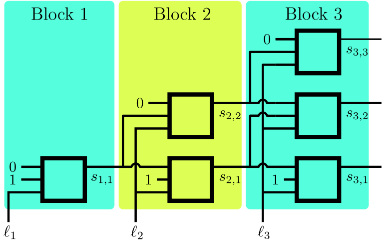
Example Pseudo-Boolean-to-CNF Translation

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

Meaning of $s_{i,j}$-variable:

$s_{i,j}$ true $\Leftrightarrow \ell_1 + \ldots + \ell_i \geq j$





$\overline{\ell}_1 \vee s_{1,1}$	$\overline{\ell}_2 \vee \overline{s}_{1,1} \vee s_{2,2}$	$\ell_3 \vee s_{2,1} \vee \overline{s}_{3,1}$	$\overline{\ell}_3 \vee \overline{s}_{2,2} \vee s_{3,3}$
$\ell_1 \vee \overline{s}_{1,1}$	$\ell_2 \vee \overline{s}_{2,2}$	$\overline{\ell}_3 \vee \overline{s}_{2,1} \vee s_{3,2}$	$\ell_3 \vee \overline{s}_{3,3}$
$\overline{\ell}_2 \vee s_{2,1}$	$s_{1,1} \vee \overline{s}_{2,2}$	$\overline{s}_{2,2} \vee s_{3,2}$	$s_{2,2} \vee \overline{s}_{3,3}$
$\overline{s}_{1,1} \vee s_{2,1}$	$\overline{\ell}_3 \vee s_{3,1}$	$\ell_3 \vee s_{2,2} \vee \overline{s}_{3,2}$	$s_{3,2}$
$\ell_2 \vee s_{1,1} \vee \overline{s}_{2,1}$	$\overline{s}_{2,1} \vee s_{3,1}$	$s_{2,1} \vee \overline{s}_{3,2}$	

Pseudo-Boolean Proof Logging with VeriPB

- Inspired by SAT proof logging, but operates on 0–1 linear constraints
- Supports efficient proof logging for
  - SAT solving — including advanced techniques previously beyond efficient proof logging like
    - Gaussian elimination [GN21]
    - Symmetry breaking [BGMN23]
  - SAT-based optimization (MaxSAT) approaches like
    - Model-improving search [VDB22]
    - Core-guided search [BBN$^+$23]
  - Subgraph problems [GMN20, GMM$^+$20]
  - Constraint programming [EGMN20, GMN22, MM23]

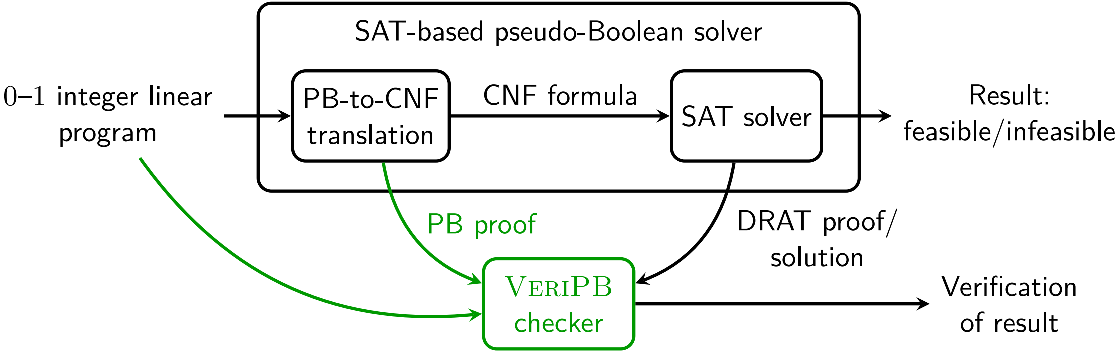
Pseudo-Boolean Proof Logging with VeriPB

- Inspired by SAT proof logging, but operates on 0–1 linear constraints
- Supports efficient proof logging for
  - SAT solving — including advanced techniques previously beyond efficient proof logging like
    - Gaussian elimination [GN21]
    - Symmetry breaking [BGMN23]
  - SAT-based optimization (MaxSAT) approaches like
    - Model-improving search [VDB22]
    - Core-guided search [BBN$^+$23]
  - Subgraph problems [GMN20, GMM$^+$20]
  - Constraint programming [EGMN20, GMN22, MM23]

This work:

- Proof logging for translating pseudo-Boolean constraints to CNF
- General framework to certify many different encodings

End-to-End Verification for Pseudo-Boolean Solving



All the Technical Details on One Slide

All the Technical Details on One Slide (or Not)

How pretty much all PB-to-CNF translations work:

1. Design circuit evaluating left-hand side of 0–1 integer linear constraint
2. Encode circuit to CNF using Tseitin translation (one new variable per circuit wire)
3. Enforce that circuit output says that the constraint is true

All the Technical Details on One Slide (or Not)

How pretty much all PB-to-CNF translations work:

1. Design circuit evaluating left-hand side of 0–1 integer linear constraint
2. Encode circuit to CNF using Tseitin translation (one new variable per circuit wire)
3. Enforce that circuit output says that the constraint is true

Important:

- We don't get to choose the translation to CNF clauses — the solver does
- We should help solver print a VeriPB proof that clauses follow from the PB constraint

All the Technical Details on One Slide (or Not)

How pretty much all PB-to-CNF translations work:

1. Design circuit evaluating left-hand side of 0–1 integer linear constraint
2. Encode circuit to CNF using Tseitin translation (one new variable per circuit wire)
3. Enforce that circuit output says that the constraint is true

Important:

- We don't get to choose the translation to CNF clauses — the solver does
- We should help solver print a VeriPB proof that clauses follow from the PB constraint

We solve this task by

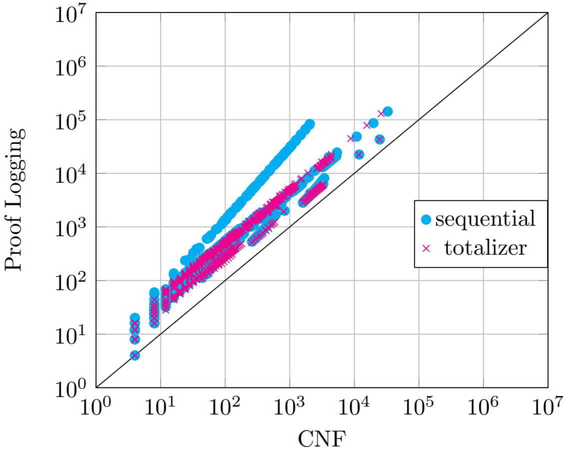
1. Understanding the circuit design
2. Deriving CNF encoding of circuit (easy)
3. Proving circuit output is true (tricky, but using pseudo-Boolean reasoning really helps)

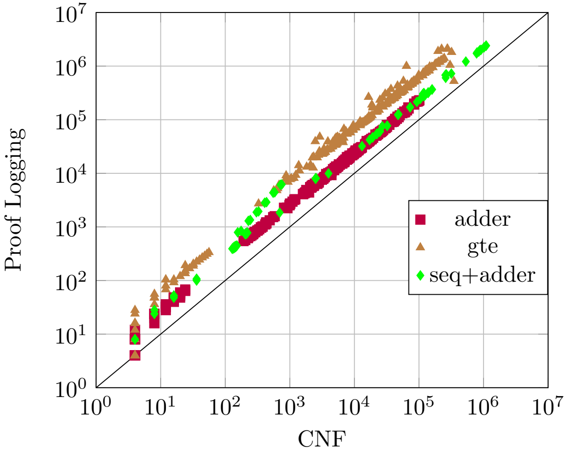
Experimental Evaluation

- Certified translations for CNF encodings with VeritasPBLib[1]
  - Sequential counter [Sin05]
  - Totalizer [BB03]
  - Generalized totalizer [JMM15]
  - Adder network [ES06]
- Proofs verified by proof checker VeriPB[2]
- Formulas solved with fork of Kissat[3] outputting VeriPB proofs
- Benchmarks from PB 2016 Evaluation[4] in 3 categories
  - 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] https://gitlab.com/MIAOresearch/tools-and-utilities/kissat_fork
[4] http://www.cril.univ-artois.fr/PB16/

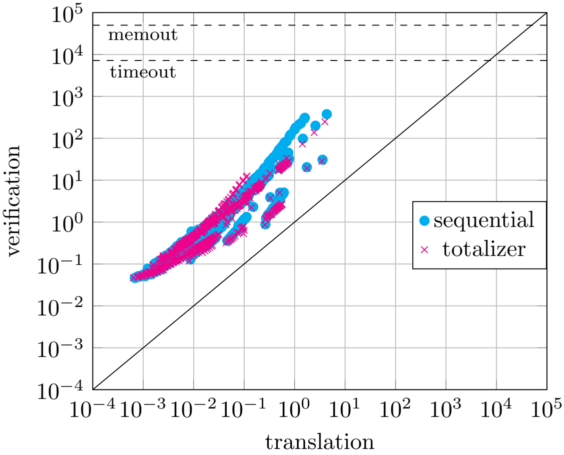
CNF Size vs Proof Size in KiB

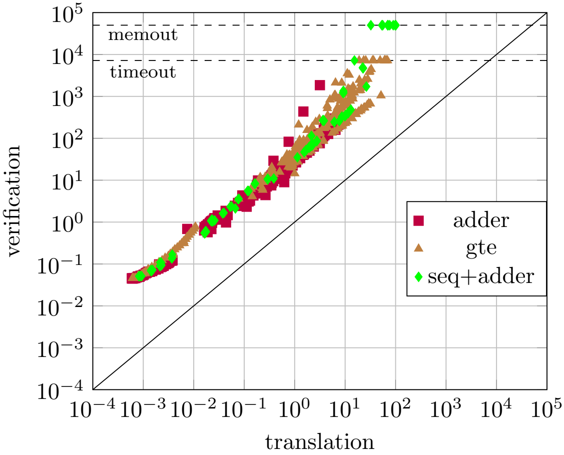




- Nice scaling for proof size in terms of original CNF formula size
- Except for some sequential encoding cases (which is not such a great encoding anyway)

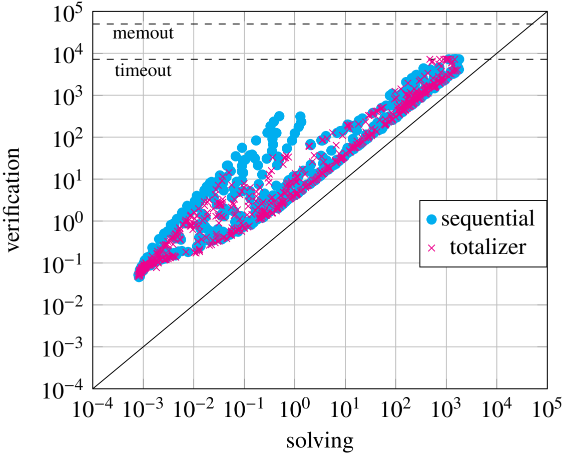
Translation Time vs Proof Checking Time in Seconds

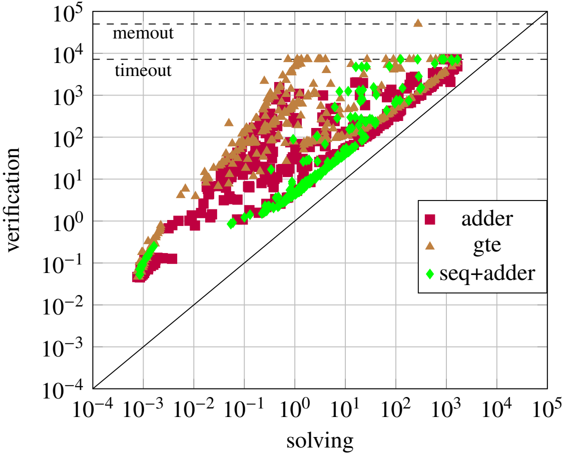




- Translation fast — only has to generate clauses and proof
- Proof checking slower — has to verify full proof

Solving Time vs Proof Checking Time in Seconds





- Room for improvement of end-to-end proof checking process
- But even first proof-of-concept implementation shows our approach is viable

Future Work

Improve performance and reliability:

- Cutting planes derivations instead of reverse unit propagations [VDB22]
- Backwards checking/trimming for verification (as in DRAT-TRIM [HHW13a])
- Fully formally verified proof checking (work in progress [BMM+23])

Extend proof logging further:

- PB-to-CNF translations with odd-even mergesort & bitonic sorting networks [Bat68]
- All of MaxSAT solving and (linear) pseudo-Boolean optimization
- Mixed integer linear programming
- Automated planning
- ...

Future Work

Improve performance and reliability:

- Cutting planes derivations instead of reverse unit propagations [VDB22]
- Backwards checking/trimming for verification (as in DRAT-TRIM [HHW13a])
- Fully formally verified proof checking (work in progress [BMM+23])

Extend proof logging further:

- PB-to-CNF translations with odd-even mergesort & bitonic sorting networks [Bat68]
- All of MaxSAT solving and (linear) pseudo-Boolean optimization
- Mixed integer linear programming
- Automated planning
- ...

We're hiring! Talk to me to join the proof logging revolution! ☺

Take-Home Message

This work:

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

Take-Home Message

This work:

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

VeriPB provides unified proof logging method for for state-of-the-art solvers in

- Boolean satisfiability (SAT) including advanced techniques [GN21, BGMN23]
- SAT-based optimization (MaxSAT) [VDB22, BBN+23]
- SAT-based pseudo-Boolean solving [**this work**]
- Constraint programming [EGMN20, GMN22, MM23]
- Subgraph solving [GMN20, GMM+20]

Take-Home Message

This work:

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

VeriPB provides unified proof logging method for for state-of-the-art solvers in

- Boolean satisfiability (SAT) including advanced techniques [GN21, BGMN23]
- SAT-based optimization (MaxSAT) [VDB22, BBN^{+}23]
- SAT-based pseudo-Boolean solving [**this work**]
- Constraint programming [EGMN20, GMN22, MM23]
- Subgraph solving [GMN20, GMM^{+}20]

Action point: What can VeriPB do for you? ☺

Take-Home Message

This work:

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

VeriPB provides unified proof logging method for for state-of-the-art solvers in

- Boolean satisfiability (SAT) including advanced techniques [GN21, BGMN23]
- SAT-based optimization (MaxSAT) [VDB22, BBN+23]
- SAT-based pseudo-Boolean solving [**this work**]
- Constraint programming [EGMN20, GMN22, MM23]
- Subgraph solving [GMN20, GMM+20]

Action point: What can VeriPB do for you? ☺

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.

[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)*, July 2023. To appear.

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

[BMM+23] Bart Bogaerts, Ciaran McCreesh, Magnus O. Myreen, Jakob Nordström, Andy Oertel, and Yong Kiam Tan. Documentation of VeriPB and CakePB for the SAT competition 2023. Available at `https://satcompetition.github.io/2023/checkers.html`, March 2023.

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.

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

References III

[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.

[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.

References IV

[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.

[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)*, August 2023. To appear.

[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.

References V

[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.

[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.