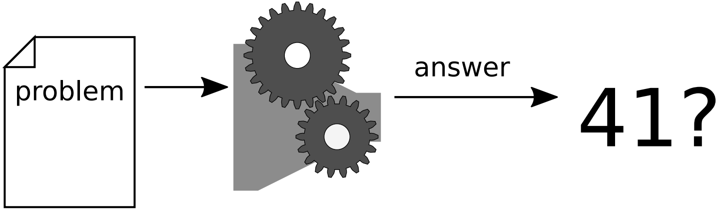
Certifying CNF Encodings of Pseudo-Boolean Constraints

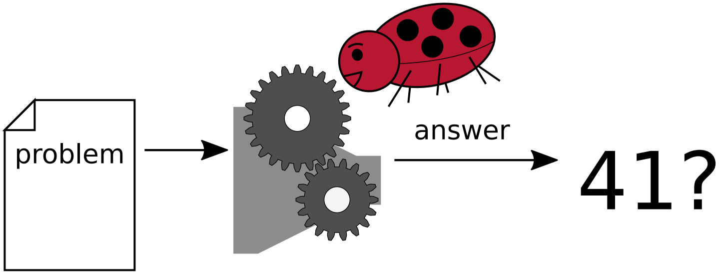
Stephan Gocht, Ruben Martins and Jakob Nordström

July 2021

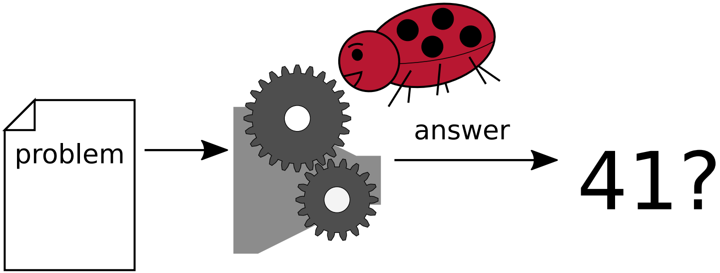
Detecting Bugs with Certifying Algorithms



Detecting Bugs with Certifying Algorithms

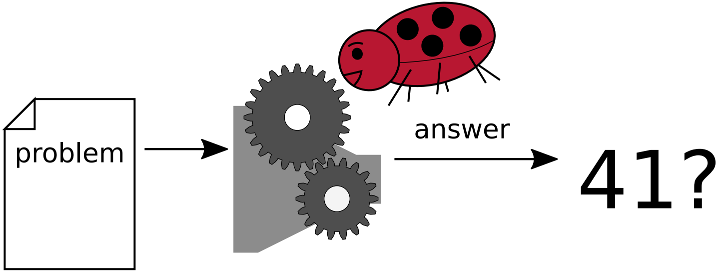


Detecting Bugs with Certifying Algorithms



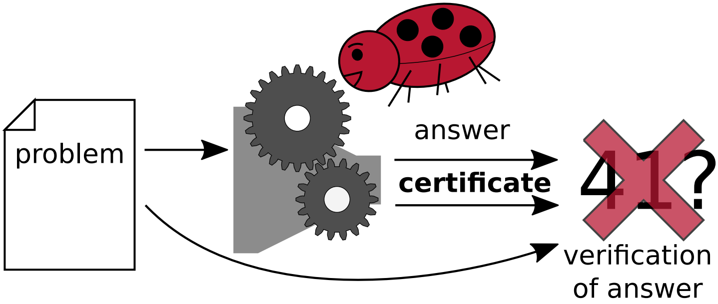
- formally verify solver?
  usually not feasible / too costly

Detecting Bugs with Certifying Algorithms



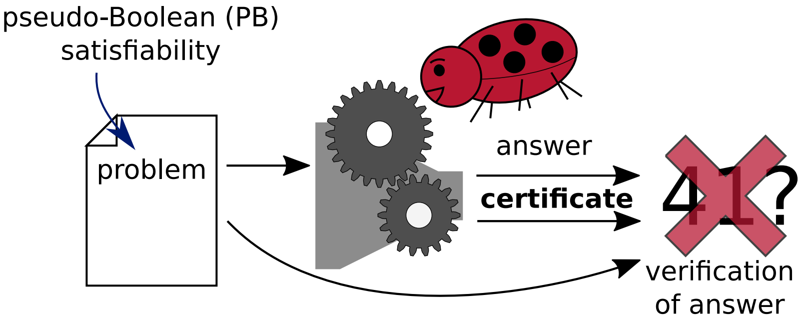
- formally verify solver?
  usually not feasible / too costly
- instead: formally verify answer!

Detecting Bugs with Certifying Algorithms



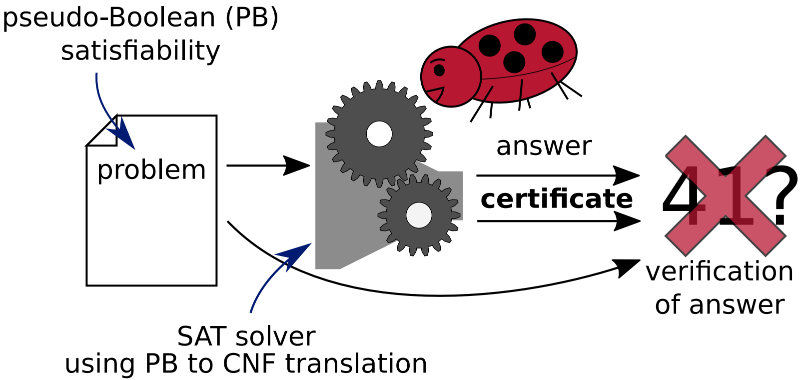
- formally verify solver?
  usually not feasible / too costly
- instead: formally verify answer!

Detecting Bugs with Certifying Algorithms



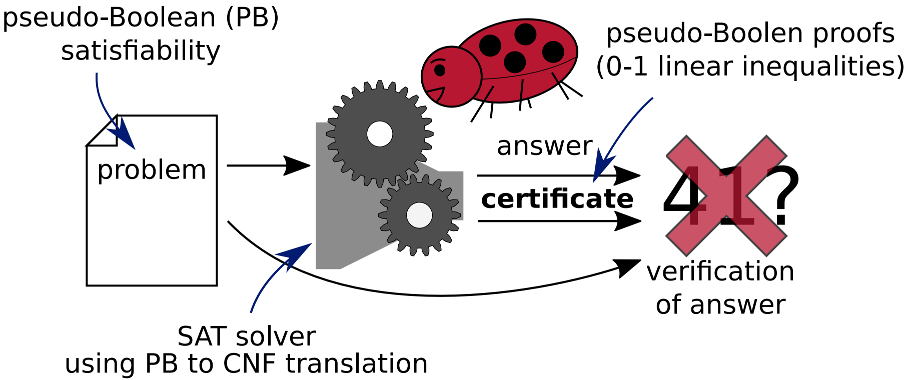
- formally verify solver?
  usually not feasible / too costly
- instead: formally verify answer!

Detecting Bugs with Certifying Algorithms



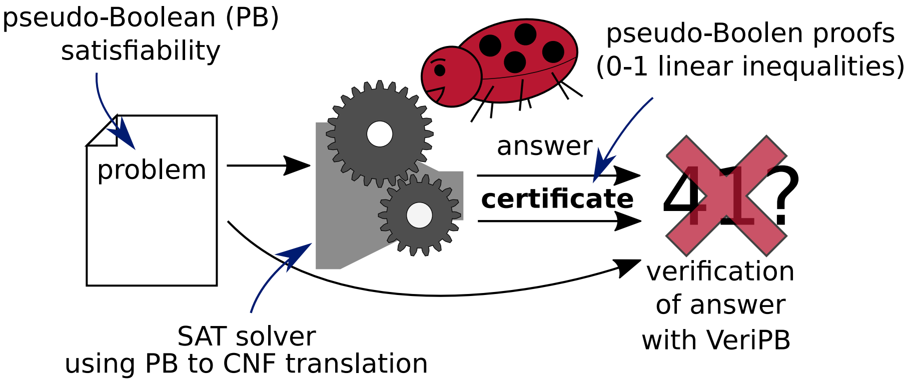
- formally verify solver?
  usually not feasible / too costly
- instead: formally verify answer!

Detecting Bugs with Certifying Algorithms



- formally verify solver?
  usually not feasible / too costly
- instead: formally verify answer!

Detecting Bugs with Certifying Algorithms



- formally verify solver?
  usually not feasible / too costly
- instead: formally verify answer!

SAT Solving — A Success Story for Certifying Algorithms . . .

- SAT = satisfiability testing of propositional formulas
- SAT competition requires solver to produce certificate (aka proof logging)

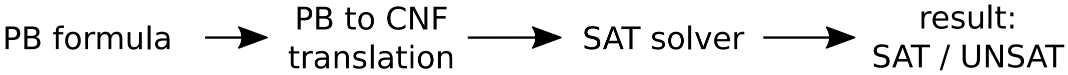
SAT Solving — A Success Story for Certifying Algorithms ...

- SAT = satisfiability testing of propositional formulas
- SAT competition requires solver to produce certificate (aka proof logging)
- Proof formats such as RUP [GN03], TraceCheck [Bie06], GRIT [CFMSSK17], LRAT [CFHH$^+$17]; DRAT [WHH14] has become standard.

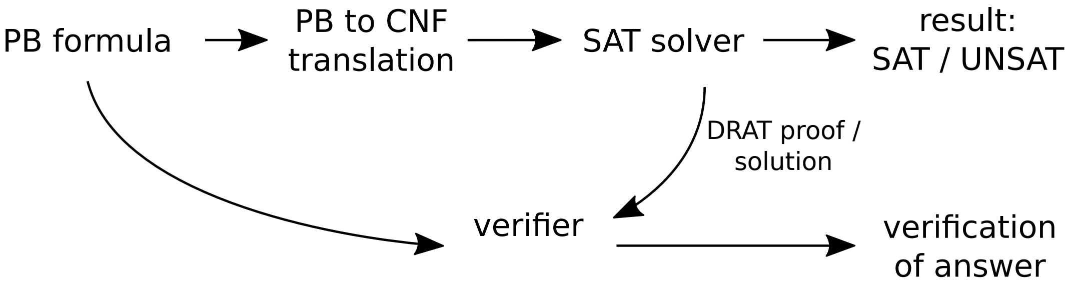
SAT Solving — A Success Story for Certifying Algorithms . . .

- SAT = satisfiability testing of propositional formulas
- SAT competition requires solver to produce certificate (aka proof logging)
- Proof formats such as RUP [GN03], TraceCheck [Bie06], GRIT [CFMSSK17], LRAT [CFHH^{+}17]; DRAT [WHH14] has become standard.
- certificates can help to
  - prove correctness of answer
  - detect and fix bugs, even when solver produced correct answer
  - audit answer later on
  - explain what solver is doing

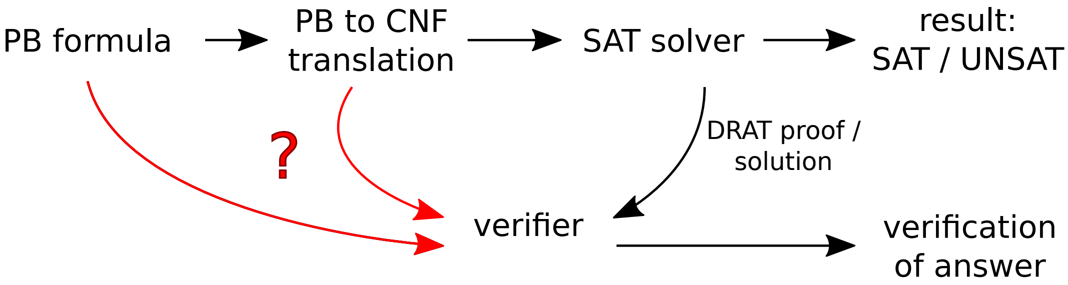
. . . That is Easy to Use . . .



. . . That is Easy to Use . . .



. . . That is Easy to Use . . .



...Except for SAT Solving Techniques That Can't Be Certified

- too much overhead / too complicated proof logging for
  - Parity reasoning (as in CryptoMiniSat [Cry] and Lingeling [Lin])
  - Counting arguments (as in Lingeling)
  - Symmetry breaking (as in BreakID [Bre])

  $\Rightarrow$ no available implementations for proof logging

- Not using these techniques $\Rightarrow$ exponential loss in reasoning power / performance

…Except for SAT Solving Techniques That Can't Be Certified

- too much overhead / too complicated proof logging for
  - Parity reasoning (as in CryptoMiniSat [Cry] and Lingeling [Lin])
  - Counting arguments (as in Lingeling)
  - Symmetry breaking (as in BreakID [Bre])

  $\Rightarrow$ no available implementations for proof logging

- Not using these techniques $\Rightarrow$ exponential loss in reasoning power / performance

- How about practical proof logging for stronger solving paradigms?
  - MaxSAT solving
  - constraint programming (CP)
  - mixed integer programming (MIP)
  - algebraic reasoning / Gröbner basis computations
  - **pseudo-Boolean satisfiablity** and optimization

New Proof Systems on the Rise

many new proof systems with implemented proof checkers:

- propagation redundancy (PR) [HKB17]
- practical polynomial calculus (PAC) [RBK18, KFB20]
- propagation redundancy for BDDs [BB21]
- Max-SAT resolution [PCH21]
- **pseudo-Boolean proofs** [EGMN20, GN21]

Our Work

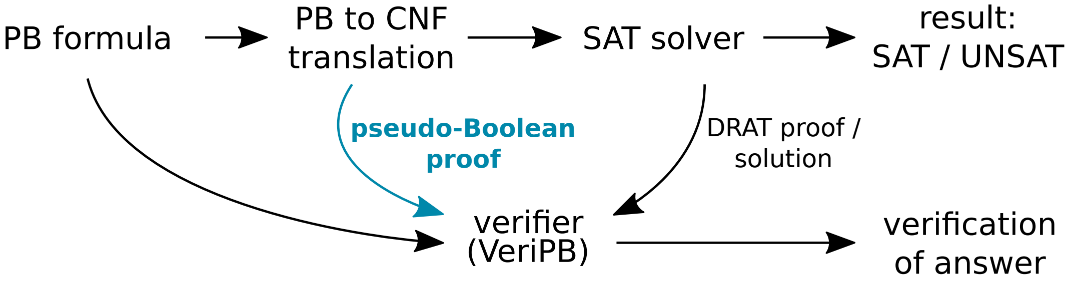
- general purpose proof format: pseudo-Boolean proofs (PBP)
- reference implementation of verifier: VeriPB[1]
- allows easy proof logging for
  - reasoning with 0-1 linear inequalities (by design)
  - all-different constraints [EGMN20]
  - subgraph isomorphism [GMN20]
  - clique and maximum common (connected) subgraph [GMM$^+$20]
  - parity reasoning [GN21]
  - SAT pre- and inprocessing [GN21]

This Work (by using VeriPB)

- proof logging for translating 0-1 linear inequalities to CNF (work in progress)
  so far only sequential counter [Sin05], many more encodings exist
- allows proof logging for SAT-based pseudo-Bolean solving

[1] `https://gitlab.com/MIAOresearch/VeriPB`

System Overview



Basic Notation

Starting from

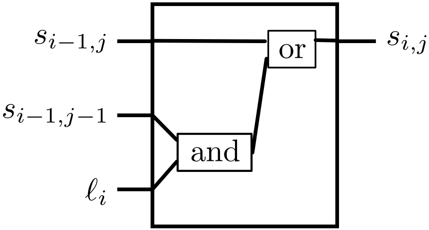
$$x_1 + x_2 + x_3 \geq 2$$

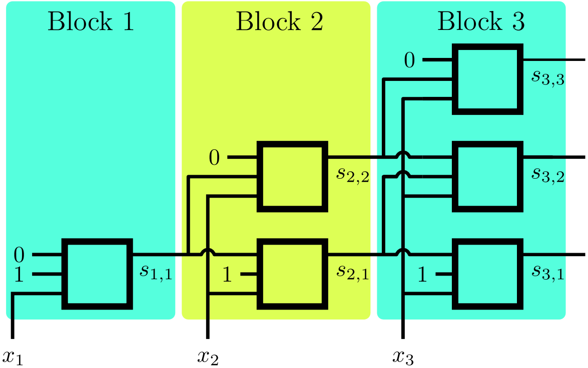
want to derive

$$\bar{x}_1 + s_{1,1} \geq 1$$
$$x_1 + \bar{s}_{1,1} \geq 1$$
$$\bar{x}_2 + s_{2,1} \geq 1$$
$$\bar{s}_{1,1} + s_{2,1} \geq 1$$
$$x_2 + s_{1,1} + \bar{s}_{2,1} \geq 1$$
$$\bar{x}_2 + \bar{s}_{1,1} + s_{2,2} \geq 1$$
$$\dots$$

- Boolean variable x with domain 0 (false) or 1 (true)
- Literal: x or its negation $\bar{x} = 1 - x$
- Pseudo-Boolean (PB) constraint: linear (in-)equality over literals
- Clause: at-least-one constraint
- Proof Format:
  - based on pseudo-Boolean constraints
  - has operations to reason with PB constraints

Sequential Counter Encoding





High level specification:

Block 1: $x_1 = s_{1,1}$

Block 2: $x_2 + s_{1,1} = s_{2,1} + s_{2,2}$ $\quad s_{2,1} \geq s_{2,2}$

Block 3: $x_3 + s_{2,1} + s_{2,2} = s_{3,1} + s_{3,2} + s_{3,3}$ $\quad s_{3,1} \geq s_{3,2} \geq s_{3,3}$

High Level Specification

Block 1:	$x_1 = s_{1,1}$	
Block 2:	$x_2 + s_{1,1} = s_{2,1} + s_{2,2}$	$s_{2,1} \geq s_{2,2}$
Block 3:	$x_3 + s_{2,1} + s_{2,2} = s_{3,1} + s_{3,2} + s_{3,3}$	$s_{3,1} \geq s_{3,2} \geq s_{3,3}$

- Specification is pseudo-Boolean!

High Level Specification

Block 1:	$x_1 = s_{1,1}$	
Block 2:	$x_2 + s_{1,1} = s_{2,1} + s_{2,2}$	$s_{2,1} \geq s_{2,2}$
Block 3:	$x_3 + s_{2,1} + s_{2,2} = s_{3,1} + s_{3,2} + s_{3,3}$	$s_{3,1} \geq s_{3,2} \geq s_{3,3}$

- Specification is pseudo-Boolean!
- $s_{i,j}$ variables are fresh, i.e., not in formula
- $\Rightarrow$ always OK to add these constraints
- can be derived in VeriPB proof system

From Specification to CNF

High Level Specification:

Block 1:	$x_1 = s_{1,1}$	
Block 2:	$x_2 + s_{1,1} = s_{2,1} + s_{2,2}$	$s_{2,1} \geq s_{2,2}$
Block 3:	$x_3 + s_{2,1} + s_{2,2} = s_{3,1} + s_{3,2} + s_{3,3}$	$s_{3,1} \geq s_{3,2} \geq s_{3,3}$

- to enforce $x_1 + x_2 + x_3 \geq 2$ need to fix output bits $s_{3,1}, s_{3,2}, s_{3,3}$

From Specification to CNF

High Level Specification:

Block 1:	$x_1 = s_{1,1}$	
Block 2:	$x_2 + s_{1,1} = s_{2,1} + s_{2,2}$	$s_{2,1} \geq s_{2,2}$
Block 3:	$x_3 + s_{2,1} + s_{2,2} = s_{3,1} + s_{3,2} + s_{3,3}$	$s_{3,1} \geq s_{3,2} \geq s_{3,3}$

- to enforce $x_1 + x_2 + x_3 \geq 2$ need to fix output bits $s_{3,1}, s_{3,2}, s_{3,3}$

  Add specifications together:

$$x_1 + x_2 + x_3 + s_{1,1} + s_{2,1} + s_{2,2} = s_{1,1} + s_{2,1} + s_{2,2} + s_{3,1} + s_{3,2} + s_{3,3}$$

From Specification to CNF

High Level Specification:

Block 1:	$x_1 = s_{1,1}$	
Block 2:	$x_2 + s_{1,1} = s_{2,1} + s_{2,2}$	$s_{2,1} \geq s_{2,2}$
Block 3:	$x_3 + s_{2,1} + s_{2,2} = s_{3,1} + s_{3,2} + s_{3,3}$	$s_{3,1} \geq s_{3,2} \geq s_{3,3}$

- to enforce $x_1 + x_2 + x_3 \geq 2$ need to fix output bits $s_{3,1}, s_{3,2}, s_{3,3}$

  Add specifications together:

$$x_1 + x_2 + x_3 + s_{1,1} + s_{2,1} + s_{2,2} = s_{1,1} + s_{2,1} + s_{2,2} + s_{3,1} + s_{3,2} + s_{3,3}$$

From Specification to CNF

High Level Specification:

Block 1:	$x_1 = s_{1,1}$	
Block 2:	$x_2 + s_{1,1} = s_{2,1} + s_{2,2}$	$s_{2,1} \geq s_{2,2}$
Block 3:	$x_3 + s_{2,1} + s_{2,2} = s_{3,1} + s_{3,2} + s_{3,3}$	$s_{3,1} \geq s_{3,2} \geq s_{3,3}$

- to enforce $x_1 + x_2 + x_3 \geq 2$ need to fix output bits $s_{3,1}, s_{3,2}, s_{3,3}$

  Add specifications together:

$$x_1 + x_2 + x_3 = s_{3,1} + s_{3,2} + s_{3,3}$$

From Specification to CNF

High Level Specification:

Block 1:	$x_1 = s_{1,1}$	
Block 2:	$x_2 + s_{1,1} = s_{2,1} + s_{2,2}$	$s_{2,1} \geq s_{2,2}$
Block 3:	$x_3 + s_{2,1} + s_{2,2} = s_{3,1} + s_{3,2} + s_{3,3}$	$s_{3,1} \geq s_{3,2} \geq s_{3,3}$

- to enforce $x_1 + x_2 + x_3 \geq 2$ need to fix output bits $s_{3,1}, s_{3,2}, s_{3,3}$

  Add specifications together:

  $$2 \leq x_1 + x_2 + x_3 \quad = \quad s_{3,1} + s_{3,2} + s_{3,3}$$

From Specification to CNF

High Level Specification:

Block 1:	$x_1 = s_{1,1}$	
Block 2:	$x_2 + s_{1,1} = s_{2,1} + s_{2,2}$	$s_{2,1} \geq s_{2,2}$
Block 3:	$x_3 + s_{2,1} + s_{2,2} = s_{3,1} + s_{3,2} + s_{3,3}$	$s_{3,1} \geq s_{3,2} \geq s_{3,3}$

- to enforce $x_1 + x_2 + x_3 \geq 2$ need to fix output bits $s_{3,1}, s_{3,2}, s_{3,3}$

  Add specifications together:

  $$2 \leq x_1 + x_2 + x_3 = s_{3,1} + s_{3,2} + s_{3,3}$$

- only used addition of constraints

From Specification to CNF

High Level Specification:

$$\begin{aligned}
&\text{Block 1:} && x_1 = s_{1,1} && \\
&\text{Block 2:} && x_2 + s_{1,1} = s_{2,1} + s_{2,2} && s_{2,1} \geq s_{2,2} \\
&\text{Block 3:} && x_3 + s_{2,1} + s_{2,2} = s_{3,1} + s_{3,2} + s_{3,3} && s_{3,1} \geq s_{3,2} \geq s_{3,3}
\end{aligned}$$

- to enforce $x_1 + x_2 + x_3 \geq 2$ need to fix output bits $s_{3,1}, s_{3,2}, s_{3,3}$

  Add specifications together:

  $$2 \leq x_1 + x_2 + x_3 = s_{3,1} + s_{3,2} + s_{3,3}$$

- only used addition of constraints
- given specification, all clauses in encoding trivial to derive (via RUP)

From Specification to CNF

High Level Specification:

Block 1:	$x_1 = s_{1,1}$	
Block 2:	$x_2 + s_{1,1} = s_{2,1} + s_{2,2}$	$s_{2,1} \geq s_{2,2}$
Block 3:	$x_3 + s_{2,1} + s_{2,2} = s_{3,1} + s_{3,2} + s_{3,3}$	$s_{3,1} \geq s_{3,2} \geq s_{3,3}$

- to enforce $x_1 + x_2 + x_3 \geq 2$ need to fix output bits $s_{3,1}, s_{3,2}, s_{3,3}$

  Add specifications together:

  $$2 \leq x_1 + x_2 + x_3 \quad = \quad s_{3,1} + s_{3,2} + s_{3,3}$$

- only used addition of constraints
- given specification, all clauses in encoding trivial to derive (via RUP)

$\Rightarrow$ derivation can be expressed in VeriPB proof system

Conclusion

- proof logging is well-established standard for SAT solving
- so far, not applicable to stronger paradigms

2

Conclusion

- proof logging is well-established standard for SAT solving
- so far, not applicable to stronger paradigms

Our work: Proof logging translating 0-1 linear inequalities to CNF with VeriPB[2]

- allows proof logging for SAT based pseudo-Boolean solving

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

Conclusion

- proof logging is well-established standard for SAT solving
- so far, not applicable to stronger paradigms

Our work: Proof logging translating 0-1 linear inequalities to CNF with VeriPB[2]

- allows proof logging for SAT based pseudo-Boolean solving

Future work:

- provide efficient proof logging also for optimization (pseudo-Boolean optimization, MaxSAT, MIP)
- express MaxSAT techniques (e.g. core guided search, MaxHS) in PB language
- new expressive proof formats and verifiers for competitions (why not with VeriPB ;-))

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

References I

[BB21] Lee A. Barnett and Armin Biere.
Non-clausal redundancy properties.
In *Proceedings of the 28th International Conference on Automated Deduction (CADE-28)*, page to appear, 2021.

[Bie06] Armin Biere.
Tracecheck.
`http://fmv.jku.at/tracecheck/`, 2006.

[Bre] BreakID.
`https://bitbucket.org/krr/breakid/src/master/`.

[CFHH+17] Luís Cruz-Filipe, Marijn J. H. Heule, Warren A. Hunt, Matt Kaufmann, and Peter Schneider-Kamp.
Efficient certified rat verification.
In Leonardo de Moura, editor, *Automated Deduction – CADE 26*, pages 220–236, Cham, 2017. Springer International Publishing.

[CFMSSK17] Luís Cruz-Filipe, Joao Marques-Silva, and Peter Schneider-Kamp.
Efficient certified resolution proof checking.
In Axel Legay and Tiziana Margaria, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 118–135. Springer Berlin Heidelberg, 2017.

References II

[Cry] CryptoMiniSat.
https://github.com/msoos/cryptominisat/.

[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. To appear*, 2020.

[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 Helmut Simonis, editor, *Principles and Practice of Constraint Programming - 26th International Conference, CP 2020, Louvain-la-Neuve, Belgium, September 7-11, 2020, Proceedings*, volume 12333 of *Lecture Notes in Computer Science*, pages 338–357. Springer, 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), July 2020. To appear*, 2020.

[GN03] Evguenii I. Goldberg and Yakov Novikov.
Verification of proofs of unsatisfiability for CNF formulas.
In *Design, Automation and Test in Europe Conference (DATE)*, pages 10886–10891. IEEE Computer Society, 2003.

References III

[GN21] Stephan Gocht and Jakob Nordström.
Certifying parity reasoning efficiently using pseudo-Boolean proofs.
In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 2021.
To appear.

[HKB17] Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere.
Short proofs without new variables.
In Leonardo de Moura, editor, *Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6-11, 2017, Proceedings*, volume 10395 of *Lecture Notes in Computer Science*, pages 130–147. Springer, 2017.

[KFB20] Daniela Kaufmann, Mathias Fleury, and Armin Biere.
The proof checkers pacheck and pastÃ¨que for the practical algebraic calculus.
In Ofer Strichman and Alexander Ivrii, editors, *Formal Methods in Computer-Aided Design, FMCAD 2020.*, volume 1, pages 264–269. TU Vienna Academic Press, 2020.

[Lin] Lingeling, Plingeling and Treengeling.
`http://fmv.jku.at/lingeling/`.

[PCH21] Matthieu Py, Mohamed Sami Cherif, and Djamal Habet.
A proof builder for max-sat.
In Chu-Min Li and Felip Manyà, editors, *Theory and Applications of Satisfiability Testing – SAT 2021*, pages 488–498, Cham, 2021. Springer International Publishing.

References IV

[RBK18] Daniela Ritirc, Armin Biere, and Manuel Kauers.
A practical polynomial calculus for arithmetic circuit verification.
In Anna M. Bigatti and Martin Brain, editors, *3rd International Workshop on Satisfiability Checking and Symbolic Computation (SC2'18)*, pages 61–76. CEUR-WS, 2018.

[Sin05] Carsten Sinz.
Towards an optimal CNF encoding of boolean cardinality constraints.
In Peter van Beek, editor, *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, volume 3709 of *Lecture Notes in Computer Science*, pages 827–831. Springer, 2005.

[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.