Certifying Without Loss of Generality Reasoning In Solution-Improving Maximum Satisfiability

Dieter Vandesande
Joint work with Jeremias Berg, Bart Bogaerts, Jakob Nordström, Andy Oertel and Tobias Paxian

September 3, 2024

ARTIFICIAL
INTELLIGENCE
RESEARCH GROUP

COMBINATORIAL SOLVING AND OPTIMIZATION

- Searching an assignment of values to variables that satisfy a set of constraints (and optimizes an objective).

COMBINATORIAL SOLVING AND OPTIMIZATION

- Searching an assignment of values to variables that satisfy a set of constraints (and optimizes an objective).
- Revolution last couple of decades in combinatorial solvers for
  - Boolean satisfiability (SAT) solving [BHvMW21]
  - Maximum Satisfiability (MaxSAT) [LM21, BJM21]
  - Satisfiability modulo theories (SMT) solving [BSST21]
  - Constraint programming (CP) [RvBW06]
  - Mixed integer linear programming (MIP) [AW13, BR07]
  - Answer Set Programming (ASP) [GKKS12]
- Solve NP problems (or worse) very successfully in practice!

COMBINATORIAL SOLVING AND OPTIMIZATION

- Searching an assignment of values to variables that satisfy a set of constraints (and optimizes an objective).
- Revolution last couple of decades in combinatorial solvers for
  - Boolean satisfiability (SAT) solving [BHvMW21]
  - Maximum Satisfiability (MaxSAT) [LM21, BJM21]
  - Satisfiability modulo theories (SMT) solving [BSST21]
  - Constraint programming (CP) [RvBW06]
  - Mixed integer linear programming (MIP) [AW13, BR07]
  - Answer Set Programming (ASP) [GKKS12]
- Solve NP problems (or worse) very successfully in practice!
- Except solvers are sometimes wrong... [BLB10, CKSW13, AGJ+18, GSD19, GS19]

COMBINATORIAL SOLVING AND OPTIMIZATION

- Searching an assignment of values to variables that satisfy a set of constraints (and optimizes an objective).
- Revolution last couple of decades in combinatorial solvers for
  - Boolean satisfiability (SAT) solving [BHvMW21]
  - Maximum Satisfiability (MaxSAT) [LM21, BJM21]
  - Satisfiability modulo theories (SMT) solving [BSST21]
  - Constraint programming (CP) [RvBW06]
  - Mixed integer linear programming (MIP) [AW13, BR07]
  - Answer Set Programming (ASP) [GKKS12]
- Solve NP problems (or worse) very successfully in practice!
- Except solvers are sometimes wrong... [BLB10, CKSW13, AGJ+18, GSD19, GS19]
- Software testing doesn't suffice to resolve this problem

COMBINATORIAL SOLVING AND OPTIMIZATION

- Searching an assignment of values to variables that satisfy a set of constraints (and optimizes an objective).
- Revolution last couple of decades in combinatorial solvers for
  - Boolean satisfiability (SAT) solving [BHvMW21]
  - Maximum Satisfiability (MaxSAT) [LM21, BJM21]
  - Satisfiability modulo theories (SMT) solving [BSST21]
  - Constraint programming (CP) [RvBW06]
  - Mixed integer linear programming (MIP) [AW13, BR07]
  - Answer Set Programming (ASP) [GKKS12]
- Solve NP problems (or worse) very successfully in practice!
- Except solvers are sometimes wrong... [BLB10, CKSW13, AGJ+18, GSD19, GS19]
- Software testing doesn't suffice to resolve this problem
- Formal verification techniques cannot deal with complexity of modern solvers [BHI+23]

CERTIFIED RESULTS WITH PROOF LOGGING

Design certifying algorithms [ABM+11, MMNS11] that

- not only solve problem but also
- do proof logging to certify that
  - the solver's **answer is correct**

CERTIFIED RESULTS WITH PROOF LOGGING

Design certifying algorithms [ABM+11, MMNS11] that

- not only solve problem but also
- do proof logging to certify that
  - the solver's **answer is correct**
  - obtained by **correct reasoning**

CERTIFIED RESULTS WITH PROOF LOGGING

Design certifying algorithms [ABM+11, MMNS11] that

- not only solve problem but also
- do proof logging to certify that
  - the solver's **answer is correct**
  - obtained by **correct reasoning**

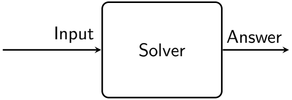
Proof logging should be done

- with **minimal overhead**
- without changing a **solver's reasoning**

CERTIFIED RESULTS WITH PROOF LOGGING

Workflow:

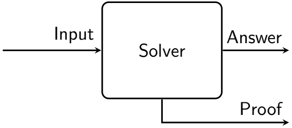
1. Run solver on problem input



CERTIFIED RESULTS WITH PROOF LOGGING

Workflow:

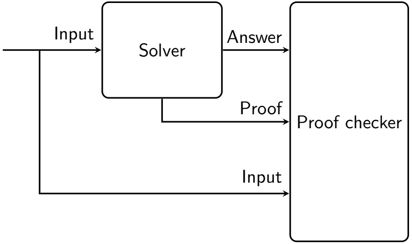
1. Run solver on problem input
2. Get as output not only an answer but also proof



CERTIFIED RESULTS WITH PROOF LOGGING

Workflow:

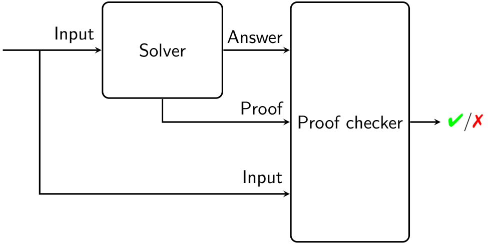
1. Run solver on problem input
2. Get as output not only an answer but also proof
3. Feed input + answer + proof to proof checker



CERTIFIED RESULTS WITH PROOF LOGGING

Workflow:

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



YET ANOTHER SAT SUCCESS STORY

Well established — required in main track of SAT competitions

YET ANOTHER SAT SUCCESS STORY

Well established — required in main track of SAT competitions

Many proof logging formats for SAT solving using CNF clausal format:

- *DRAT* [HHW13a, HHW13b, WHH14]
- *GRIT* [CMS17]
- *LRAT* [CHH+17]
- ...

YET ANOTHER SAT SUCCESS STORY

Well established — required in main track of SAT competitions

Many proof logging formats for SAT solving using CNF clausal format:

- *DRAT* [HHW13a, HHW13b, WHH14]
- *GRIT* [CMS17]
- *LRAT* [CHH+17]
- ...

Formally verified proof checkers exist

YET ANOTHER SAT SUCCESS STORY

Well established — required in main track of SAT competitions

Many proof logging formats for SAT solving using CNF clausal format:

- *DRAT* [HHW13a, HHW13b, WHH14]
- *GRIT* [CMS17]
- *LRAT* [CHH+17]
- ...

Formally verified proof checkers exist

But efficient proof logging has remained out of reach for other paradigms, e.g. Maximum Satisfiability (MaxSAT)

OUTLINE OF THIS PRESENTATION

- MaxSAT and how to certify it
- Pacose and its intricate without-loss-of-generality reasoning
- An introduction on the VeriPB proof system
- Proof logging Pacose
- Conclusions & Future work

OUTLINE OF THIS PRESENTATION

- MaxSAT and how to certify it
- Pacose and its intricate without-loss-of-generality reasoning
- An introduction on the VeriPB proof system
- Proof logging Pacose
- Conclusions & Future work

PRELIMINARIES

Example:
$$F = \{x_1 \vee x_2,\ x_2 \vee x_3,\ x_1 \vee \overline{x_2} \vee x_3\}$$

- Boolean variable: x
- Assignment α: assigns variables true $(= 1)$ or false $(= 0)$
- Literal l: variable x (satisfied if $\alpha(x) = 1$) or its negation $\overline{x}$ (satisfied if $\alpha(x) = 0$)
- Clause C: Disjunction of literals $l_1 \vee \cdots \vee l_k$
  (C is satisfied by α if at least one literal in C is assigned true)
- Propositional formula in CNF: $F = C_1 \wedge \cdots \wedge C_n$
  (F is satisfied if all clauses C_i are satisfied)

THE MAXIMUM SATISFIABILITY PROBLEM

Example:

$$F = \{x_1 \vee x_2,\ x_2 \vee x_3,\ x_1 \vee \overline{x_2} \vee x_3\}$$
$$\mathcal{O} = x_1 + x_2 + x_3$$

Optimization variant of Satisfiability Problem.

A MaxSAT-instance is a tuple $(F, \mathcal{O})$ with:

- F a propositional formula
- $\mathcal{O}$ an integer linear objective over Boolean variables

THE MAXIMUM SATISFIABILITY PROBLEM

Example:

$F = \{x_1 \vee x_2,\ x_2 \vee x_3,\ x_1 \vee \overline{x_2} \vee x_3\}$
$\mathcal{O} = x_1 + x_2 + x_3$
Solution: $\alpha = \{x_1 \mapsto 1, x_2 \mapsto 0, x_3 \mapsto 1\}$

Optimization variant of Satisfiability Problem.

A MaxSAT-instance is a tuple $(F, \mathcal{O})$ with:

- F a propositional formula
- $\mathcal{O}$ an integer linear objective over Boolean variables

A (feasible) solution is an assignment for all variables such that F is satisfied.

THE MAXIMUM SATISFIABILITY PROBLEM

Example:

$F = \{x_1 \vee x_2,\ x_2 \vee x_3,\ x_1 \vee \overline{x_2} \vee x_3\}$

$\mathcal{O} = x_1 + x_2 + x_3$

Solution: $\alpha = \{x_1 \mapsto 1, x_2 \mapsto 0, x_3 \mapsto 1\}$

Optimization variant of Satisfiability Problem.

A MaxSAT-instance is a tuple $(F, \mathcal{O})$ with:

- F a propositional formula
- $\mathcal{O}$ an integer linear objective over Boolean variables

A (feasible) solution is an assignment for all variables such that F is satisfied.

An optimal solution is a solution such that no other solution has higher objective value.

PROOF SYSTEMS FOR MAXSAT REASONING

Proof systems for MaxSAT are studied theoretically for proof complexity

- MaxSAT resolution [LH05, HL06, BLM06, BLM07]
- Tableaux reasoning [LMS16, LCH+22, LM22]
- Cost-aware redundancy notions [BMM13, BJ19, IBJ22]

PROOF SYSTEMS FOR MAXSAT REASONING

Proof systems for MaxSAT are studied theoretically for proof complexity

- MaxSAT resolution [LH05, HL06, BLM06, BLM07]
- Tableaux reasoning [LMS16, LCH+22, LM22]
- Cost-aware redundancy notions [BMM13, BJ19, IBJ22]

Solvers specifically designed for emitting proofs

- MaxSAT resolution [PCH21, PCH22]
- Cost Resolution [LNOR11]

PROOF SYSTEMS FOR MAXSAT REASONING

Proof systems for MaxSAT are studied theoretically for proof complexity

- MaxSAT resolution [LH05, HL06, BLM06, BLM07]
- Tableaux reasoning [LMS16, LCH+22, LM22]
- Cost-aware redundancy notions [BMM13, BJ19, IBJ22]

Solvers specifically designed for emitting proofs

- MaxSAT resolution [PCH21, PCH22]
- Cost Resolution [LNOR11]

No certified state-of-the-art MaxSAT solver using native proof system!

MAXSAT SOLVERS

Four main categories:

- Branch-and-Bound
- Solution-Improving
- Core-Guided
- Implicit Hitting Set

Different reasoning techniques!

CERTIFIED MAXSAT SOLVERS

1st idea (Does not work):

- Utilize one of SAT's proof systems

CERTIFIED MAXSAT SOLVERS

1st idea (Does not work):

- Utilize one of SAT's proof systems

  Inherently not able to reason about optimality

CERTIFIED MAXSAT SOLVERS

1^{st} idea (Does not work):

- Utilize one of SAT's proof systems

  Inherently not able to reason about optimality

2^{nd} idea (Does not work):

- Obtain solution α with $\mathcal{O}(\alpha) = v^*$ for $(F, \mathcal{O})$ by running MaxSAT solver

CERTIFIED MAXSAT SOLVERS

1st idea (Does not work):

- Utilize one of SAT's proof systems
  Inherently not able to reason about optimality

2nd idea (Does not work):

- Obtain solution α with $\mathcal{O}(\alpha) = v^*$ for $(F, \mathcal{O})$ by running MaxSAT solver
- Check solution to be satisfying assignment

CERTIFIED MAXSAT SOLVERS

1st idea (Does not work):

- Utilize one of SAT's proof systems
  Inherently not able to reason about optimality

2nd idea (Does not work):

- Obtain solution α with $\mathcal{O}(\alpha) = v^*$ for $(F, \mathcal{O})$ by running MaxSAT solver
- Check solution to be satisfying assignment

- Create formula $F' = F \wedge \quad \mathcal{O} > v^*$

CERTIFIED MAXSAT SOLVERS

1st idea (Does not work):

- Utilize one of SAT's proof systems
  Inherently not able to reason about optimality

2nd idea (Does not work):

- Obtain solution α with $\mathcal{O}(\alpha) = v^*$ for $(F, \mathcal{O})$ by running MaxSAT solver
- Check solution to be satisfying assignment
- Create formula $F' = F \wedge \quad \mathcal{O} > v^*$
- Run SAT solver with standard proof logging to obtain certificate of UNSAT for F'

CERTIFIED MAXSAT SOLVERS

1^{st} idea (Does not work):

- Utilize one of SAT's proof systems

  Inherently not able to reason about optimality

2^{nd} idea (Does not work):

- Obtain solution α with $\mathcal{O}(\alpha) = v^*$ for $(F, \mathcal{O})$ by running MaxSAT solver
- Check solution to be satisfying assignment

  Easy to check!
- Create formula $F' = F \wedge \quad \mathcal{O} > v^*$
- Run SAT solver with standard proof logging to obtain certificate of UNSAT for F'

CERTIFIED MAXSAT SOLVERS

1[st] idea (Does not work):

- Utilize one of SAT's proof systems
  Inherently not able to reason about optimality

2[nd] idea (Does not work):

- Obtain solution α with $\mathcal{O}(\alpha) = v^*$ for $(F, \mathcal{O})$ by running MaxSAT solver
- Check solution to be satisfying assignment
  Easy to check!
- Create formula $F' = F \wedge \mathrm{CNF}(\mathcal{O} > v^*)$
  Requires proof logging – Not possible with state-of-the-art proof systems for SAT
- Run SAT solver with standard proof logging to obtain certificate of UNSAT for F'

CERTIFIED MAXSAT SOLVERS

1^{st} idea (Does not work):

- Utilize one of SAT's proof systems
  Inherently not able to reason about optimality

2^{nd} idea (Does not work):

- Obtain solution α with $\mathcal{O}(\alpha) = v^*$ for $(F, \mathcal{O})$ by running MaxSAT solver
- Check solution to be satisfying assignment
  Easy to check!
- Create formula $F' = F \wedge \mathrm{CNF}(\mathcal{O} > v^*)$
  Requires proof logging – Not possible with state-of-the-art proof systems for SAT
- Run SAT solver with standard proof logging to obtain certificate of UNSAT for F'
  Causes serious overhead

CERTIFIED MAXSAT SOLVERS

1st idea (Does not work):

- Utilize one of SAT's proof systems

  Inherently not able to reason about optimality

2nd idea (Does not work):

- Obtain solution α with $\mathcal{O}(\alpha) = v^*$ for $(F, \mathcal{O})$ by running MaxSAT solver
- Check solution to be satisfying assignment

  Easy to check!
- Create formula $F' = F \wedge \mathrm{CNF}(\mathcal{O} > v^*)$

  Requires proof logging – Not possible with state-of-the-art proof systems for SAT
- Run SAT solver with standard proof logging to obtain certificate of UNSAT for F'

  Causes serious overhead

Only proves answer correct, not reasoning within solver!

CERTIFIED MAXSAT SOLVERS

3rd idea:

- Express the solver's reasoning in a more general proof system

CERTIFIED MAXSAT SOLVERS

3^{rd} idea:

- Express the solver's reasoning in a more general proof system
  VeriPB!

CERTIFIED MAXSAT SOLVERS

3rd idea:

- Express the solver's reasoning in a more general proof system
  VeriPB!

A small and recent history of VeriPB MaxSAT proof logging:

CERTIFIED MAXSAT SOLVERS

3rd idea:

- Express the solver's reasoning in a more general proof system
  VeriPB!

A small and recent history of VeriPB MaxSAT proof logging:

- QMaxSAT: Solution Improving Search [Van23, VDB22]
  - Focus on certifying PB-to-CNF encodings

CERTIFIED MAXSAT SOLVERS

3rd idea:

- Express the solver's reasoning in a more general proof system
  VeriPB!

A small and recent history of VeriPB MaxSAT proof logging:

- QMaxSAT: Solution Improving Search [Van23, VDB22]
  - Focus on certifying PB-to-CNF encodings
- RC2 and CGSS: Core-Guided Search [BBN+23]
  - Including techniques such as stratification, hardening, intrinsic-at-most-ones constraints, ...

CERTIFIED MAXSAT SOLVERS

3rd idea:

- Express the solver's reasoning in a more general proof system
  VeriPB!

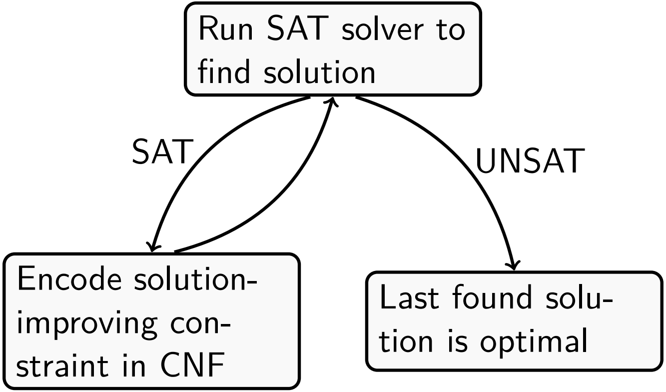
A small and recent history of VeriPB MaxSAT proof logging:

- QMaxSAT: Solution Improving Search [Van23, VDB22]
  - Focus on certifying PB-to-CNF encodings
- RC2 and CGSS: Core-Guided Search [BBN+23]
  - Including techniques such as stratification, hardening, intrinsic-at-most-ones constraints, ...
- This paper – Pacose: State-Of-The-Art Solution Improving Search
  - Challenge: without-loss-of-generality reasoning in the Dynamic Polynomial Watchdog encoding

OUTLINE OF THIS PRESENTATION

- MaxSAT and how to certify it
- Pacose and its intricate without-loss-of-generality reasoning
- An introduction on the VeriPB proof system
- Proof logging Pacose
- Conclusions & Future work

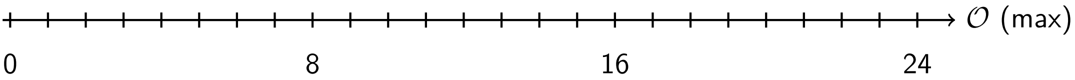
SOLUTION-IMPROVING SEARCH



HOW PACOSE WORKS

Introduction of variables:

- $\mathcal{Z}$ output-variables with $\mathrm{CNF}(z_k \leftrightarrow \mathcal{O} \geq k \cdot 2^p + T)$
- $\mathcal{T}$ representing a value $T = \sum_{i=0}^{p-1} 2^i t_i$ with $0 \leq T \leq 2^p - 1$ and t_i fresh variables



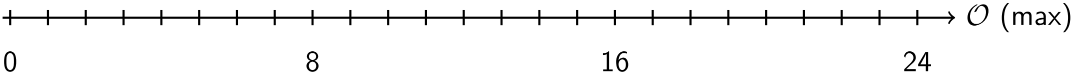
HOW PACOSE WORKS

Introduction of variables:

- $\mathcal{Z}$ output-variables with $\text{CNF}(z_k \leftrightarrow \mathcal{O} \geq k \cdot 2^p + T)$
- $\mathcal{T}$ representing a value $T = \sum_{i=0}^{p-1} 2^i t_i$ with $0 \leq T \leq 2^p - 1$ and t_i fresh variables

Coarse Convergence: Search increasingly for interval containing optimal value by only playing with $\mathcal{Z}$-variables

Fine Convergence: Play with value of T to find actual optimal value.



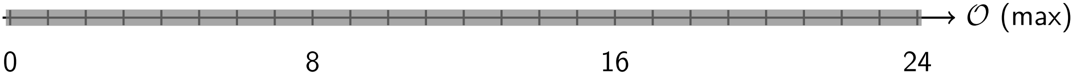
HOW PACOSE WORKS

Introduction of variables:

- $\mathcal{Z}$ output-variables with $\text{CNF}(z_k \leftrightarrow \mathcal{O} \geq k \cdot 2^p + T)$
- $\mathcal{T}$ representing a value $T = \sum_{i=0}^{p-1} 2^i t_i$ with $0 \leq T \leq 2^p - 1$ and t_i fresh variables

Coarse Convergence: Search increasingly for interval containing optimal value by only playing with $\mathcal{Z}$-variables

Fine Convergence: Play with value of T to find actual optimal value.



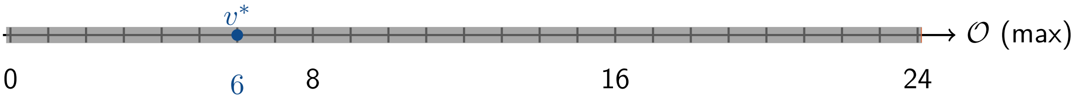
HOW PACOSE WORKS

Introduction of variables:

- $\mathcal{Z}$ output-variables with $\mathrm{CNF}(z_k \leftrightarrow \mathcal{O} \geq k \cdot 2^p + T)$
- $\mathcal{T}$ representing a value $T = \sum_{i=0}^{p-1} 2^i t_i$ with $0 \leq T \leq 2^p - 1$ and t_i fresh variables

Coarse Convergence: Search increasingly for interval containing optimal value by only playing with $\mathcal{Z}$-variables

Fine Convergence: Play with value of T to find actual optimal value.



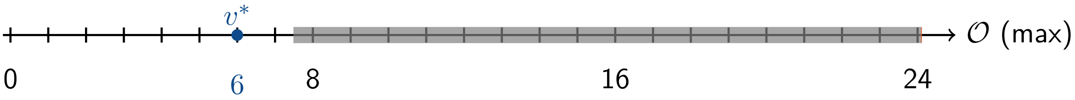
HOW PACOSE WORKS

Introduction of variables:

- $\mathcal{Z}$ output-variables with $\mathrm{CNF}(z_k \leftrightarrow \mathcal{O} \geq k \cdot 2^p + T)$
- $\mathcal{T}$ representing a value $T = \sum_{i=0}^{p-1} 2^i t_i$ with $0 \leq T \leq 2^p - 1$ and t_i fresh variables

Coarse Convergence: Search increasingly for interval containing optimal value by only playing with $\mathcal{Z}$-variables

Fine Convergence: Play with value of T to find actual optimal value.



$$z_1 \leftrightarrow \mathcal{O} \geq 1 \cdot 8 + T$$

Assume $z_1 \geq 1$

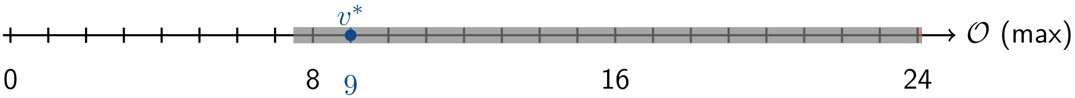
HOW PACOSE WORKS

Introduction of variables:

- $\mathcal{Z}$ output-variables with $\text{CNF}(z_k \leftrightarrow \mathcal{O} \geq k \cdot 2^p + T)$
- $\mathcal{T}$ representing a value $T = \sum_{i=0}^{p-1} 2^i t_i$ with $0 \leq T \leq 2^p - 1$ and t_i fresh variables

Coarse Convergence: Search increasingly for interval containing optimal value by only playing with $\mathcal{Z}$-variables

Fine Convergence: Play with value of T to find actual optimal value.



$$z_1 \leftrightarrow \mathcal{O} \geq 1 \cdot 8 + T$$

Assume $z_1 \geq 1$

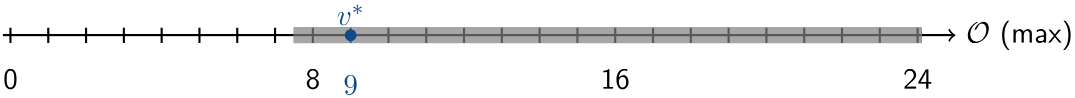
HOW PACOSE WORKS

Introduction of variables:

- $\mathcal{Z}$ output-variables with $\mathrm{CNF}(z_k \leftrightarrow \mathcal{O} \geq k \cdot 2^p + T)$
- $\mathcal{T}$ representing a value $T = \sum_{i=0}^{p-1} 2^i t_i$ with $0 \leq T \leq 2^p - 1$ and t_i fresh variables

Coarse Convergence: Search increasingly for interval containing optimal value by only playing with $\mathcal{Z}$-variables

Fine Convergence: Play with value of T to find actual optimal value.



$$z_1 \leftrightarrow \mathcal{O} \geq 1 \cdot 8 + T$$
$$z_1 \geq 1 \text{ (W.L.O.G. } T = 0)$$

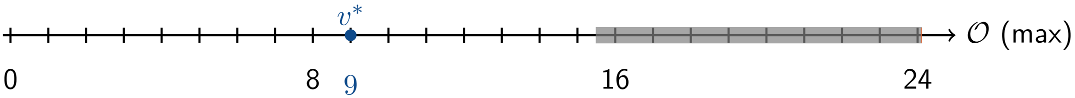
HOW PACOSE WORKS

Introduction of variables:

- $\mathcal{Z}$ output-variables with $\text{CNF}(z_k \leftrightarrow \mathcal{O} \geq k \cdot 2^p + T)$
- $\mathcal{T}$ representing a value $T = \sum_{i=0}^{p-1} 2^i t_i$ with $0 \leq T \leq 2^p - 1$ and t_i fresh variables

Coarse Convergence: Search increasingly for interval containing optimal value by only playing with $\mathcal{Z}$-variables

Fine Convergence: Play with value of T to find actual optimal value.



$z_2 \leftrightarrow \mathcal{O} \geq 2 \cdot 8 + T$

Assume $z_2 \geq 1$

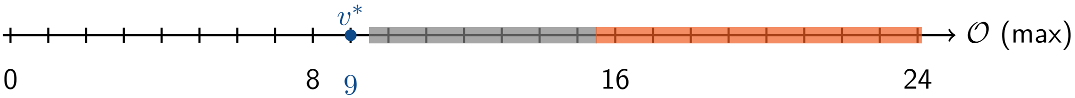
HOW PACOSE WORKS

Introduction of variables:

- $\mathcal{Z}$ output-variables with $\text{CNF}(z_k \leftrightarrow \mathcal{O} \geq k \cdot 2^p + T)$
- $\mathcal{T}$ representing a value $T = \sum_{i=0}^{p-1} 2^i t_i$ with $0 \leq T \leq 2^p - 1$ and t_i fresh variables

Coarse Convergence: Search increasingly for interval containing optimal value by only playing with $\mathcal{Z}$-variables

Fine Convergence: Play with value of T to find actual optimal value.



$$z_2 \leftrightarrow \mathcal{O} \geq 2 \cdot 8 + T$$
$$\overline{z_2} \geq 1$$

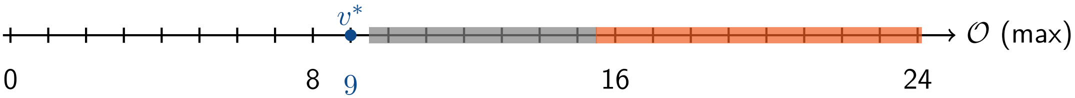
HOW PACOSE WORKS

Introduction of variables:

- $\mathcal{Z}$ output-variables with $\text{CNF}(z_k \leftrightarrow \mathcal{O} \geq k \cdot 2^p + T)$
- $\mathcal{T}$ representing a value $T = \sum_{i=0}^{p-1} 2^i t_i$ with $0 \leq T \leq 2^p - 1$ and t_i fresh variables

Coarse Convergence: Search increasingly for interval containing optimal value by only playing with $\mathcal{Z}$-variables

Fine Convergence: Play with value of T to find actual optimal value.



$$z_1 \leftrightarrow \mathcal{O} \geq 1 \cdot 8 + T$$

Previously found: $z_1 \geq 1$

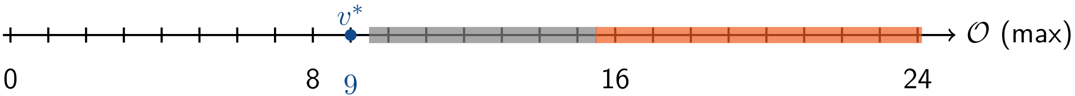
HOW PACOSE WORKS

Introduction of variables:

- $\mathcal{Z}$ output-variables with $\mathrm{CNF}(z_k \leftrightarrow \mathcal{O} \geq k \cdot 2^p + T)$
- $\mathcal{T}$ representing a value $T = \sum_{i=0}^{p-1} 2^i t_i$ with $0 \leq T \leq 2^p - 1$ and t_i fresh variables

Coarse Convergence: Search increasingly for interval containing optimal value by only playing with $\mathcal{Z}$-variables

Fine Convergence: Play with value of T to find actual optimal value.



$$\mathcal{O} \geq 1 \cdot 8 + T$$

Previously found: $z_1 \geq 1$

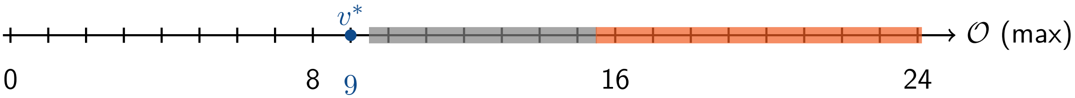
HOW PACOSE WORKS

Introduction of variables:

- $\mathcal{Z}$ output-variables with $\text{CNF}(z_k \leftrightarrow \mathcal{O} \geq k \cdot 2^p + T)$
- $\mathcal{T}$ representing a value $T = \sum_{i=0}^{p-1} 2^i t_i$ with $0 \leq T \leq 2^p - 1$ and t_i fresh variables

Coarse Convergence: Search increasingly for interval containing optimal value by only playing with $\mathcal{Z}$-variables

Fine Convergence: Play with value of T to find actual optimal value.



$$\mathcal{O} \geq 1 \cdot 8 + T$$

Assume $T \geq 2$

$$T = t_0 + 2t_1 + 4t_2$$

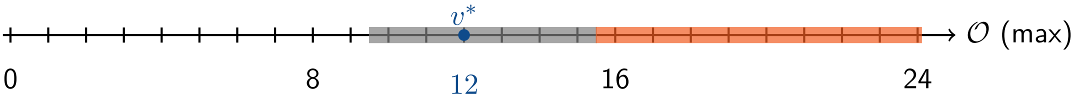
HOW PACOSE WORKS

Introduction of variables:

- $\mathcal{Z}$ output-variables with $\mathrm{CNF}(z_k \leftrightarrow \mathcal{O} \geq k \cdot 2^p + T)$
- $\mathcal{T}$ representing a value $T = \sum_{i=0}^{p-1} 2^i t_i$ with $0 \leq T \leq 2^p - 1$ and t_i fresh variables

Coarse Convergence: Search increasingly for interval containing optimal value by only playing with $\mathcal{Z}$-variables

Fine Convergence: Play with value of T to find actual optimal value.



$\mathcal{O} \geq 1 \cdot 8 + T$

Assume $T \geq 2$

$T = t_0 + 2t_1 + 4t_2$

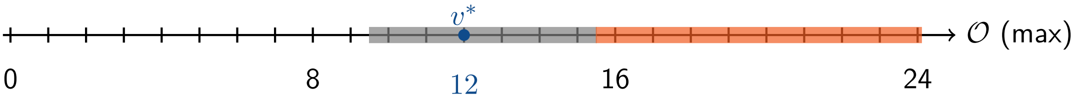
HOW PACOSE WORKS

Introduction of variables:

- $\mathcal{Z}$ output-variables with $\text{CNF}(z_k \leftrightarrow \mathcal{O} \geq k \cdot 2^p + T)$
- $\mathcal{T}$ representing a value $T = \sum_{i=0}^{p-1} 2^i t_i$ with $0 \leq T \leq 2^p - 1$ and t_i fresh variables

Coarse Convergence: Search increasingly for interval containing optimal value by only playing with $\mathcal{Z}$-variables

Fine Convergence: Play with value of T to find actual optimal value.



$$\mathcal{O} \geq 1 \cdot 8 + T$$

$T = t_0 + 2t_1 + 4t_2$ $\quad$ $t_2 \geq 1$ since $T \geq 4$ (W.L.O.G. $T = 4$)

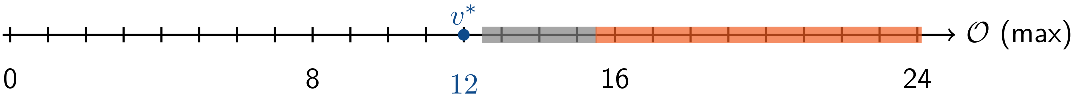
HOW PACOSE WORKS

Introduction of variables:

- $\mathcal{Z}$ output-variables with $\mathrm{CNF}(z_k \leftrightarrow \mathcal{O} \geq k \cdot 2^p + T)$
- $\mathcal{T}$ representing a value $T = \sum_{i=0}^{p-1} 2^i t_i$ with $0 \leq T \leq 2^p - 1$ and t_i fresh variables

Coarse Convergence: Search increasingly for interval containing optimal value by only playing with $\mathcal{Z}$-variables

Fine Convergence: Play with value of T to find actual optimal value.



$\mathcal{O} \geq 1 \cdot 8 + T$

Assume $T \geq 5$

$T = t_0 + 2t_1 + 4t_2$

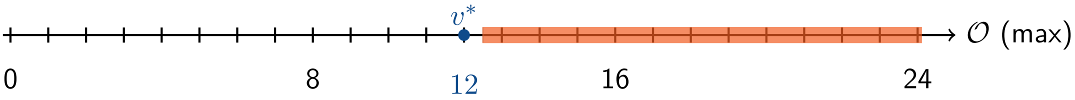
HOW PACOSE WORKS

Introduction of variables:

- $\mathcal{Z}$ output-variables with $\mathrm{CNF}(z_k \leftrightarrow \mathcal{O} \geq k \cdot 2^p + T)$
- $\mathcal{T}$ representing a value $T = \sum_{i=0}^{p-1} 2^i t_i$ with $0 \leq T \leq 2^p - 1$ and t_i fresh variables

Coarse Convergence: Search increasingly for interval containing optimal value by only playing with $\mathcal{Z}$-variables

Fine Convergence: Play with value of T to find actual optimal value.



$\mathcal{O} \geq 1 \cdot 8 + T$

Assume $T \geq 5$

$T = t_0 + 2t_1 + 4t_2$

SUSPICIOUS DERIVATIONS?

Without loss of generality:

- Coarse Convergence: repeatedly use that wlog $T = 0$
- Fine Convergence: use that wlog $T = n$ for increasing n

Sounds about right?

SUSPICIOUS DERIVATIONS?

Without loss of generality:

- Coarse Convergence: repeatedly use that wlog $T = 0$
- Fine Convergence: use that wlog $T = n$ for increasing n

Sounds about right? How to fit this in formal proof system?

SUSPICIOUS DERIVATIONS?

Without loss of generality:

- Coarse Convergence: repeatedly use that wlog $T = 0$
- Fine Convergence: use that wlog $T = n$ for increasing n

Sounds about right? How to fit this in formal proof system?

The proof system VeriPB guarantees that if

- we first derive $z_1 \geq 1$ using wlog $T = 0$,
- we later derive $T \geq 4$ using wlog $T = 4$,

the second derivation will have a proof obligation that $z_1 \geq 1$ remains to hold if $T = 4$.

OUTLINE OF THIS PRESENTATION

- MaxSAT and how to certify it
- Pacose and its intricate without-loss-of-generality reasoning
- An introduction on the VeriPB proof system
- Proof logging Pacose
- Conclusions & Future work

VeriPB: A PROOF SYSTEM FOR PSEUDO-BOOLEAN OPTIMIZATION

VeriPB is a proof system for pseudo-Boolean optimization [BGMN22, EGMN20].

A pseudo-Boolean constraint is a 0–1 integer linear inequalities:

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

- $a_i, A \in \mathbb{Z}$
- literals ℓ_i: x_i or $\overline{x}_i$ (where $x_i + \overline{x}_i = 1$)

SOME TYPES OF PSEUDO-BOOLEAN CONSTRAINTS

1. Clauses

$$x_1 \vee \overline{x}_2 \vee x_3 \quad \Leftrightarrow \quad x_1 + \overline{x}_2 + x_3 \geq 1$$

2. Cardinality constraints

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

3. General pseudo-Boolean constraints

$$x_1 + 2\overline{x}_2 + 3x_3 + 4\overline{x}_4 + 5x_5 \geq 7$$

REASONING OVER PSEUDO-BOOLEAN CONSTRAINTS USING *VeriPB*

VeriPB reasons on such pseudo-Boolean constraints with:

REASONING OVER PSEUDO-BOOLEAN CONSTRAINTS USING *VeriPB*

VeriPB reasons on such pseudo-Boolean constraints with:

- pseudo-Boolean reasoning with the Cutting Planes proof system [CCT87]
  - e.g., adding up two constraints

REASONING OVER PSEUDO-BOOLEAN CONSTRAINTS USING *VeriPB*

VeriPB reasons on such pseudo-Boolean constraints with:

- pseudo-Boolean reasoning with the Cutting Planes proof system [CCT87]
  - e.g., adding up two constraints
- Redundance-Based Strenghtening [GN21, BGMN22]
  - generalisation of the RAT-rule [BT19]
  - allows introducing "fresh" reification variables, such as $r \Leftrightarrow (\sum_i a_i l_i \geq A)$

REASONING OVER PSEUDO-BOOLEAN CONSTRAINTS USING *VeriPB*

VeriPB reasons on such pseudo-Boolean constraints with:

- pseudo-Boolean reasoning with the Cutting Planes proof system [CCT87]
  - e.g., adding up two constraints
- Redundance-Based Strenghtening [GN21, BGMN22]
  - generalisation of the RAT-rule [BT19]
  - allows introducing "fresh" reification variables, such as $r \Leftrightarrow (\sum_i a_i l_i \geq A)$
- Support for Optimisation [BGMN22]
  - allows deriving model-improving constraints $(\mathcal{O} > v^*)$
  - proving optimality by contradiction

WRITING DERIVATIONS TO A PROOF FILE

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

WRITING DERIVATIONS TO A PROOF FILE

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

Naming constraints by integers and literal axioms by the literal involved (with $\sim$ for negation) as

$$\text{Constraint 1} \doteq w + 2x + y \geq 2$$
$$\text{Constraint 2} \doteq w + 2x + 4y + 3z \geq 5$$

WRITING DERIVATIONS TO A PROOF FILE

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

Naming constraints by integers and literal axioms by the literal involved (with $\sim$ for negation) as

$$\begin{aligned} \text{Constraint 1} &\doteq w + 2x + y \geq 2 \\ \text{Constraint 2} &\doteq w + 2x + 4y + 3z \geq 5 \end{aligned}$$

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

```
pol 1 2 * 2 + 3 d
```

OUTLINE OF THIS PRESENTATION

- MaxSAT and how to certify it
- Pacose and its intricate without-loss-of-generality reasoning
- An introduction on the VeriPB proof system
- Proof logging Pacose
- Conclusions & Future work

PROOF LOGGING PACOSE

By reification: $z_1 \rightarrow \mathcal{O} \geq 8 + T$. In pseudo-Boolean, this is

$$16\overline{z_1} + \mathcal{O} \geq 8 + T \tag{1}$$

By without-loss-of-generality reasoning:

$$z_1 \geq 1 \tag{2}$$

PROOF LOGGING PACOSE

By reification: $z_1 \rightarrow \mathcal{O} \geq 8 + T$. In pseudo-Boolean, this is

$$16\overline{z_1} + \mathcal{O} \geq 8 + T \tag{1}$$

By without-loss-of-generality reasoning:

$$z_1 \geq 1 \tag{2}$$

Multiplying (2) by 16 results in

$$16z_1 \geq 16 \tag{3}$$

PROOF LOGGING PACOSE

By reification: $z_1 \rightarrow \mathcal{O} \geq 8 + T$. In pseudo-Boolean, this is

$$16\overline{z_1} + \mathcal{O} \geq 8 + T \tag{1}$$

By without-loss-of-generality reasoning:

$$z_1 \geq 1 \tag{2}$$

Multiplying (2) by 16 results in

$$16 z_1 \geq 16 \tag{3}$$

Addition of (1) and (3) results in

$$16z + 16\overline{z} + \mathcal{O} \geq 8 + T + 16 \tag{4}$$

PROOF LOGGING PACOSE

By reification: $z_1 \rightarrow \mathcal{O} \geq 8 + T$. In pseudo-Boolean, this is

$$16\overline{z_1} + \mathcal{O} \geq 8 + T \tag{1}$$

By without-loss-of-generality reasoning:

$$z_1 \geq 1 \tag{2}$$

Multiplying (2) by 16 results in

$$16z_1 \geq 16 \tag{3}$$

Addition of (1) and (3) results in

$$16 + \mathcal{O} \geq 8 + T + 16 \tag{4}$$

PROOF LOGGING PACOSE

By reification: $z_1 \rightarrow \mathcal{O} \geq 8 + T$. In pseudo-Boolean, this is

$$16\overline{z_1} + \mathcal{O} \geq 8 + T \tag{1}$$

By without-loss-of-generality reasoning:

$$z_1 \geq 1 \tag{2}$$

Multiplying (2) by 16 results in

$$16z_1 \geq 16 \tag{3}$$

Addition of (1) and (3) results in

$$\mathcal{O} \geq 8 + T \tag{4}$$

PROVING WITHOUT LOSS OF GENERALITY REASONING IN VERIPB

Without loss of generality:

- Coarse Convergence: derive $z_1 \geq 1$ using wlog $T = 0$,
- Fine Convergence: derive $T \geq 4$ using wlog $T = 4$,

PROVING WITHOUT LOSS OF GENERALITY REASONING IN VERIPB

Without loss of generality:

- Coarse Convergence: derive $z_1 \geq 1$ using wlog $T = 0$,
- Fine Convergence: derive $T \geq 4$ using wlog $T = 4$,

Will use redundance-based strengthening. General form: F and $F \wedge C$ equi-optimal if

$$F \wedge \neg C \models F \upharpoonright_\omega \wedge C \upharpoonright_\omega \wedge (\mathcal{O} \upharpoonright_\omega \geq \mathcal{O})$$

with ω is a substitution (replacing variables by literals or truth values).

PROVING WITHOUT LOSS OF GENERALITY REASONING IN VERIPB

Without loss of generality:

- Coarse Convergence: derive $z_1 \geq 1$ using wlog $T = 0$,
- Fine Convergence: derive $T \geq 4$ using wlog $T = 4$,

Will use redundance-based strengthening. General form: F and $F \wedge C$ equi-optimal if

$$F \wedge \neg C \models F\restriction_\omega \wedge C\restriction_\omega \wedge (\mathcal{O}\restriction_\omega \geq \mathcal{O})$$

with ω is a substitution (replacing variables by literals or truth values).

Intuition: for any assignment α that satisfies F but violates C, we show that the assignment $\alpha \circ \omega$ satisfies both F and C and has an at least as good objective value.

PROVING WITHOUT LOSS OF GENERALITY REASONING IN VERIPB

Without loss of generality:

- Coarse Convergence: derive $z_1 \geq 1$ using wlog $T = 0$,
- Fine Convergence: derive $T \geq 4$ using wlog $T = 4$,

Will use redundance-based strengthening. General form: F and $F \wedge C$ equi-optimal if

$$F \wedge \neg C \models F\restriction_\omega \wedge C\restriction_\omega \wedge (\mathcal{O}\restriction_\omega \geq \mathcal{O})$$

with ω is a substitution (replacing variables by literals or truth values).

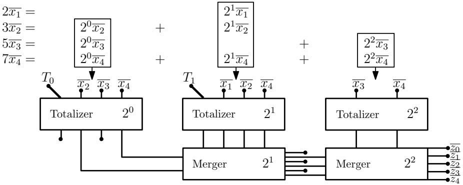
Intuition: for any assignment α that satisfies F but violates C, we show that the assignment $\alpha \circ \omega$ satisfies both F and C and has an at least as good objective value.

In our case:

- ω sets T to 4.

PROVING WITHOUT LOSS OF GENERALITY REASONING IN VERIPB (2)

Setting $T = 4$ breaks circuit $\mathcal{C}(\mathcal{O}, T)$ defining $\mathrm{CNF}(z_k \leftrightarrow \mathcal{O} \geq k \cdot 2^p + T)$!

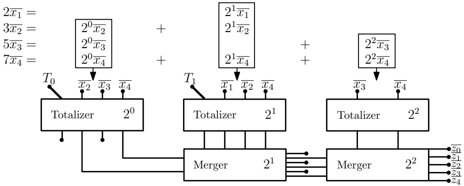


PROVING WITHOUT LOSS OF GENERALITY REASONING IN VERIPB (2)

Setting $T = 4$ breaks circuit $\mathcal{C}(\mathcal{O}, T)$ defining $\text{CNF}(z_k \leftrightarrow \mathcal{O} \geq k \cdot 2^p + T)$!

Redundance-based strengthening:

$$F \wedge \mathcal{C}(\mathcal{O}, T) \wedge \neg C \models F \upharpoonright_{\{T \mapsto 4\}} \wedge \mathcal{C}(\mathcal{O}, T) \upharpoonright_{\{T \mapsto 4\}} \wedge C \upharpoonright_{\{T \mapsto 4\}} \wedge (\mathcal{O} \upharpoonright_{\{T \mapsto 4\}} \geq \mathcal{O})$$

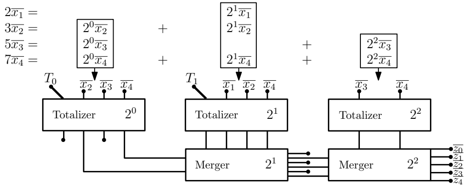


PROVING WITHOUT LOSS OF GENERALITY REASONING IN VERIPB (2)

Setting $T = 4$ breaks circuit $\mathcal{C}(\mathcal{O}, T)$ defining $\text{CNF}(z_k \leftrightarrow \mathcal{O} \geq k \cdot 2^p + T)$!

Redundance-based strengthening:

$$F \wedge \mathcal{C}(\mathcal{O}, T) \wedge \neg C \models F \upharpoonright_{\{T \mapsto 4\}} \; \wedge \mathcal{C}(\mathcal{O}, 4) \; \wedge \; C \upharpoonright_{\{T \mapsto 4\}} \; \wedge (\mathcal{O} \upharpoonright_{\{T \mapsto 4\}} \geq \mathcal{O})$$

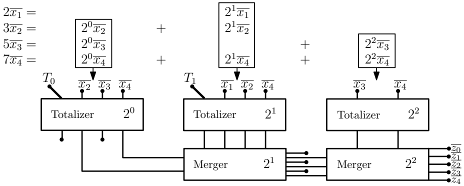


PROVING WITHOUT LOSS OF GENERALITY REASONING IN VERIPB (2)

Setting $T = 4$ breaks circuit $\mathcal{C}(\mathcal{O}, T)$ defining $\mathrm{CNF}(z_k \leftrightarrow \mathcal{O} \geq k \cdot 2^p + T)$!

Redundance-based strengthening:

$$F \land \mathcal{C}(\mathcal{O}, T) \land \neg C \models F \restriction_{\{T \mapsto 4\}} \land \mathcal{C}(\mathcal{O}, 4) \land C \restriction_{\{T \mapsto 4\}} \land (\mathcal{O} \restriction_{\{T \mapsto 4\}} \geq \mathcal{O})$$



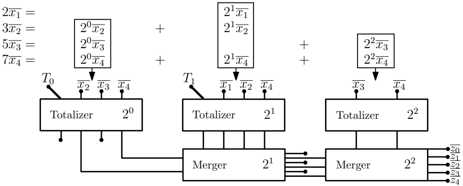
PROVING WITHOUT LOSS OF GENERALITY REASONING IN VERIPB (2)

Setting $T = 4$ breaks circuit $\mathcal{C}(\mathcal{O}, T)$ defining $\mathrm{CNF}(z_k \leftrightarrow \mathcal{O} \geq k \cdot 2^p + T)$!

Redundance-based strengthening:

$$F \wedge \mathcal{C}(\mathcal{O}, T) \wedge \neg C \models F \upharpoonright_{\{T \mapsto 4\}} \wedge \mathcal{C}(\mathcal{O}, 4) \wedge C \upharpoonright_{\{T \mapsto 4\}} \wedge (\mathcal{O} \upharpoonright_{\{T \mapsto 4\}} \geq \mathcal{O})$$

Problem: We need more expressive substitutions.



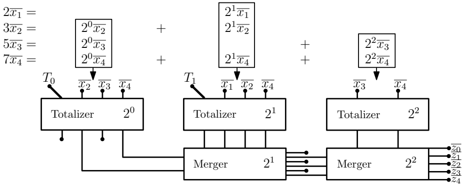
PROVING WITHOUT LOSS OF GENERALITY REASONING IN VERIPB (2)

Setting $T = 4$ breaks circuit $\mathcal{C}(\mathcal{O}, T)$ defining $\mathrm{CNF}(z_k \leftrightarrow \mathcal{O} \geq k \cdot 2^p + T)$!

Redundance-based strengthening:

$$F \wedge \mathcal{C}(\mathcal{O}, T) \wedge \neg C \models F \restriction_{\{T \mapsto 4\}} \wedge \mathcal{C}(\mathcal{O}, 4) \wedge C \restriction_{\{T \mapsto 4\}} \wedge (\mathcal{O} \restriction_{\{T \mapsto 4\}} \geq \mathcal{O})$$

Problem: We need more expressive substitutions.
Solution: Shadow Circuits (see paper)



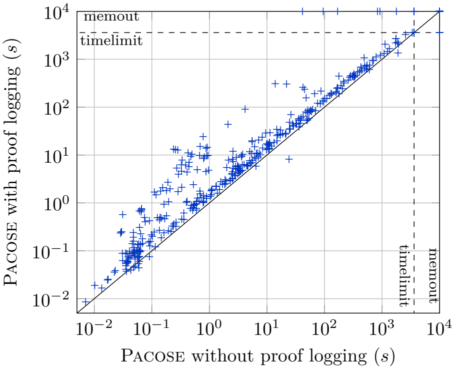
EXPERIMENTAL RESULTS

Implementation contains: TrimMaxSAT, Hardening, Binary Adder encoding

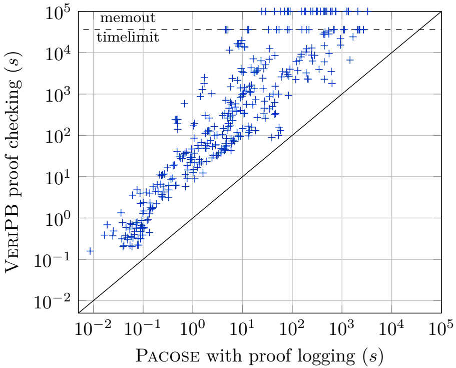
Benchmark: MaxSAT Eval. 2023 (weighted)

Resource Limits: Pacose (1h, 14GB) — VeriPB (10h, 14GB)

2/685 OoT, 9/685 OoM



29/674 OoT, 53/674 OoM



OUTLINE OF THIS PRESENTATION

- MaxSAT and how to certify it
- Pacose and its intricate without-loss-of-generality reasoning
- An introduction on the VeriPB proof system
- Proof logging Pacose
- Conclusions & Future work

FUTURE RESEARCH

Performance enhancement:

- Overhead in the solving time.
- Overhead in the checking time.

FUTURE RESEARCH

Performance enhancement:

- Overhead in the solving time.
- Overhead in the checking time.

Other MaxSAT Algorithms

- Branch-and-Bound solvers with clause learning [LXC+21]
- Implicit hitting sets solvers [DB11]

SUMMING UP

In this paper:

- Redundance-Based Strengthening can be used to proof log without-loss-of-generality reasoning in the Dynamic-Polynomial Watchdog ,
- Shadow Circuits for more expressive substitutions (without changing the proof system!)

SUMMING UP

In this paper:

- Redundance-Based Strengthening can be used to proof log without-loss-of-generality reasoning in the Dynamic-Polynomial Watchdog ,
- Shadow Circuits for more expressive substitutions (without changing the proof system!)

Proof logging helps:

- Ensuring correctness of a result.
- But also provides insights in how a solver really works.

SUMMING UP

In this paper:

- Redundance-Based Strengthening can be used to proof log without-loss-of-generality reasoning in the Dynamic-Polynomial Watchdog ,
- Shadow Circuits for more expressive substitutions (without changing the proof system!)

Proof logging helps:

- Ensuring correctness of a result.
- But also provides insights in how a solver really works.

Certifying MaxSAT solvers is viable with VeriPB proof system.

SUMMING UP

In this paper:

- Redundance-Based Strengthening can be used to proof log without-loss-of-generality reasoning in the Dynamic-Polynomial Watchdog ,
- Shadow Circuits for more expressive substitutions (without changing the proof system!)

Proof logging helps:

- Ensuring correctness of a result.
- But also provides insights in how a solver really works.

Certifying MaxSAT solvers is viable with VeriPB proof system.

Thank you for your attention!

REFERENCES

[ABM+11] Eyad Alkassar, Sascha Böhme, Kurt Mehlhorn, Christine Rizkallah, and Pascal Schweitzer. An introduction to certifying algorithms. *it - Information Technology Methoden und innovative Anwendungen der Informatik und Informationstechnik*, 53(6):287–293, December 2011.

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

[AW13] Tobias Achterberg and Roland Wunderling. Mixed integer programming: Analyzing 12 years of progress. In Michael Jünger and Gerhard Reinelt, editors, *Facets of Combinatorial Optimization*, pages 449–481. Springer, 2013.

[BBN+23] Jeremias Berg, Bart Bogaerts, Jakob Nordström, Andy Oertel, and Dieter Vandesande. Certified core-guided MaxSAT solving. In Brigitte Pientka and Cesare Tinelli, editors, *Automated Deduction - CADE 29 - 29th International Conference on Automated Deduction, Rome, Italy, July 1-4, 2023, Proceedings*, volume 14132 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2023.

REFERENCES

[BGMN22] Bart Bogaerts, Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. Certified symmetry and dominance breaking for combinatorial optimisation. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI '22)*, 2022. accepted.

[BHI+23] Tomáš Balyo, Marijn Heule, Markus Iser, Matti Järvisalo, and Martin Suda. The 2023 international SAT competition. `https://satcompetition.github.io/2023/`, 2023.

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

[BHvW21] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2021.

[BJ19] Jeremias Berg and Matti Järvisalo. Unifying reasoning and core-guided search for maximum satisfiability. In Francesco Calimeri, Nicola Leone, and Marco Manna, editors, *Logics in Artificial Intelligence - 16th European Conference, JELIA 2019, Rende, Italy, May 7-11, 2019, Proceedings*, volume 11468 of *Lecture Notes in Computer Science*, pages 287–303. Springer, 2019.

REFERENCES

[BJM21] Fahiem Bacchus, Matti Järvisalo, and Ruben Martins. Maximum satisfiabiliy. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 929–991. IOS Press, 2021.

[BLB10] Robert Brummayer, Florian Lonsing, and Armin Biere. Automated testing and debugging of SAT and QBF solvers. In *Proceedings of the 13th International Conference on Theory and Applications of Satisfiability Testing (SAT '10)*, volume 6175 of *Lecture Notes in Computer Science*, pages 44–57. Springer, July 2010.

[BLM06] Maria Luisa Bonet, Jordi Levy, and Felip Manyà. A complete calculus for max-sat. In Armin Biere and Carla P. Gomes, editors, *Theory and Applications of Satisfiability Testing - SAT 2006, 9th International Conference, Seattle, WA, USA, August 12-15, 2006, Proceedings*, volume 4121 of *Lecture Notes in Computer Science*, pages 240–251. Springer, 2006.

[BLM07] Maria Luisa Bonet, Jordi Levy, and Felip Manyà. Resolution for max-sat. *Artif. Intell.*, 171(8-9):606–618, 2007.

REFERENCES

[BMM13] Anton Belov, António Morgado, and João Marques-Silva. Sat-based preprocessing for maxsat. In Kenneth L. McMillan, Aart Middeldorp, and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning - 19th International Conference, LPAR-19, Stellenbosch, South Africa, December 14-19, 2013. Proceedings*, volume 8312 of *Lecture Notes in Computer Science*, pages 96–111. Springer, 2013.

[BR07] Robert Bixby and Edward Rothberg. Progress in computational mixed integer programming—A look back from the other side of the tipping point. *Annals of Operations Research*, 149(1):37–41, February 2007.

[BSST21] Clark W. Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. Satisfiability modulo theories. In Biere et al. [BHvW21], pages 1267–1329.

[BT19] Samuel R. Buss and Neil Thapen. DRAT proofs, propagation redundancy, and extended resolution. In *Proceedings of the 22nd International Conference on Theory and Applications of Satisfiability Testing (SAT '19)*, volume 11628 of *Lecture Notes in Computer Science*, pages 71–89. Springer, July 2019.

[CCT87] William Cook, Collette Rene Coullard, and György Turán. On the complexity of cutting-plane proofs. *Discrete Applied Mathematics*, 18(1):25–38, November 1987.

REFERENCES

[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.

[CKSW13] William Cook, Thorsten Koch, Daniel E. Steffy, and Kati Wolter. A hybrid branch-and-bound approach for exact rational mixed-integer programming. *Mathematical Programming Computation*, 5(3):305–344, September 2013.

[CMS17] Luís Cruz-Filipe, João P. Marques-Silva, and Peter Schneider-Kamp. Efficient certified resolution proof checking. In *Proceedings of the 23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS '17)*, volume 10205 of *Lecture Notes in Computer Science*, pages 118–135. Springer, April 2017.

[DB11] Jessica Davies and Fahiem Bacchus. Solving MAXSAT by solving a sequence of simpler SAT instances. In *Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming (CP '11)*, volume 6876 of *Lecture Notes in Computer Science*, pages 225–239. Springer, September 2011.

REFERENCES

[EGMN20] Jan Elffers, Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. Justifying all differences using pseudo-Boolean reasoning. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 1486–1494. AAAI Press, 2020.

[GKKS12] Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. *Answer Set Solving in Practice.* Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[GN21] Stephan Gocht and Jakob Nordström. Certifying parity reasoning efficiently using pseudo-Boolean proofs. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 3768–3777. AAAI Press, 2021.

[GS19] Graeme Gange and Peter Stuckey. Certifying optimality in constraint programming. Presentation at KTH Royal Institute of Technology. Slides available at `https://www.kth.se/polopoly_fs/1.879851.1550484700!/CertifiedCP.pdf`, February 2019.

REFERENCES

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

[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.

[HL06] Federico Heras and Javier Larrosa. New inference rules for efficient max-sat solving. In *Proceedings, The Twenty-First National Conference on Artificial Intelligence and the Eighteenth Innovative Applications of Artificial Intelligence Conference, July 16-20, 2006, Boston, Massachusetts, USA*, pages 68–73. AAAI Press, 2006.

REFERENCES

[IBJ22] Hannes Ihalainen, Jeremias Berg, and Matti Järvisalo. Clause redundancy and preprocessing in maximum satisfiability. In Jasmin Blanchette, Laura Kovács, and Dirk Pattinson, editors, *Automated Reasoning - 11th International Joint Conference, IJCAR 2022, Haifa, Israel, August 8-10, 2022, Proceedings*, volume 13385 of *Lecture Notes in Computer Science*, pages 75–94. Springer, 2022.

[LCH+22] Shoulin Li, Jordi Coll, Djamal Habet, Chu-Min Li, and Felip Manyà. A tableau calculus for maxsat based on resolution. In Atia Cortés, Francisco Grimaldo, and Tommaso Flaminio, editors, *Artificial Intelligence Research and Development - Proceedings of the 24th International Conference of the Catalan Association for Artificial Intelligence, CCIA 2022, Sitges, Spain, 19-21 October 2022*, volume 356 of *Frontiers in Artificial Intelligence and Applications*, pages 35–44. IOS Press, 2022.

[LH05] Javier Larrosa and Federico Heras. Resolution in max-sat and its relation to local consistency in weighted csps. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 193–198. Professional Book Center, 2005.

[LM21] Chu Min Li and Felip Manyà. MaxSAT, hard and soft constraints. In Biere et al. [BHvW21], pages 903–927.

REFERENCES

[LM22] Chu Min Li and Felip Manyà. Inference in maxsat and minsat. In Wolfgang Ahrendt, Bernhard Beckert, Richard Bubel, and Einar Broch Johnsen, editors, *The Logic of Software. A Tasting Menu of Formal Methods - Essays Dedicated to Reiner Hähnle on the Occasion of His 60th Birthday*, volume 13360 of *Lecture Notes in Computer Science*, pages 350–369. Springer, 2022.

[LMS16] Chu Min Li, Felip Manyà, and Joan Ramon Soler. A clause tableau calculus for minsat. In Àngela Nebot, Xavier Binefa, and Ramón López de Mántaras, editors, *Artificial Intelligence Research and Development - Proceedings of the 19th International Conference of the Catalan Association for Artificial Intelligence, Barcelona, Catalonia, Spain, October 19-21, 2016*, volume 288 of *Frontiers in Artificial Intelligence and Applications*, pages 88–97. IOS Press, 2016.

[LNOR11] Javier Larrosa, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell. A framework for certified Boolean branch-and-bound optimization. *J. Autom. Reason.*, 46(1):81–102, 2011.

[LXC+21] Chu-Min Li, Zhenxing Xu, Jordi Coll, Felip Manyà, Djamal Habet, and Kun He. Combining clause learning and branch and bound for maxsat. In *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

REFERENCES

[MMNS11] Ross M. McConnell, Kurt Mehlhorn, Stefan Näher, and Pascal Schweitzer. Certifying algorithms. *Computer Science Review*, 5(2):119–161, May 2011.

[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 - 24th International Conference, Barcelona, Spain, July 5-9, 2021, Proceedings*, volume 12831 of *Lecture Notes in Computer Science*, pages 488–498. Springer, 2021.

[PCH22] Matthieu Py, Mohamed Sami Cherif, and Djamal Habet. Proofs and certificates for max-sat. *J. Artif. Intell. Res.*, 75:1373–1400, 2022.

[RvBW06] Francesca Rossi, Peter van Beek, and Toby Walsh, editors. *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*. Elsevier, 2006.

[Van23] Dieter Vandesande. Towards certified MaxSAT solving: Certified MaxSAT solving with SAT oracles and encodings of pseudo-Boolean constraints. Master's thesis, Vrije Universiteit Brussel (VUB), 2023.

REFERENCES

[VDB22] Dieter Vandesande, Wolf De Wulf, and Bart Bogaerts. QMaxSATpb: A certified MaxSAT solver. In Georg Gottlob, Daniela Inclezan, and Marco Maratea, editors, *Logic Programming and Nonmonotonic Reasoning - 16th International Conference, LPNMR 2022, Genova, Italy, September 5-9, 2022, Proceedings*, volume 13416 of *Lecture Notes in Computer Science*, pages 429–442. Springer, 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.