Certified Branch-and-Bound MaxSAT Solving

Dieter Vandesande
Joint work with Jordi Coll, Chu-Min Li, and Bart Bogaerts

November 7, 2024
Lund

VUB
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
  - **solution 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
  - **solution is correct**
  - obtained by **correct reasoning**

CERTIFIED RESULTS WITH PROOF LOGGING

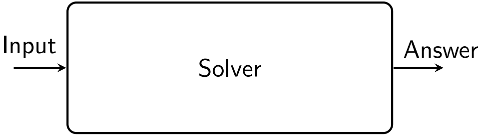
Design certifying algorithms [ABM+11, MMNS11] that

- not only solve problem but also
- do proof logging to certify that
  - **solution is correct**
  - obtained by **correct reasoning**

Proof logging should be done

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

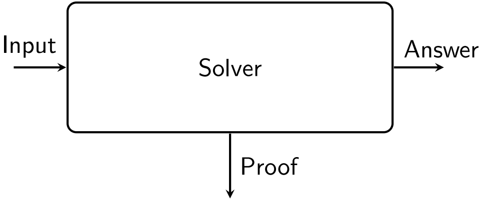
CERTIFIED RESULTS WITH PROOF LOGGING



Workflow:

1. Run solver on problem input

CERTIFIED RESULTS WITH PROOF LOGGING



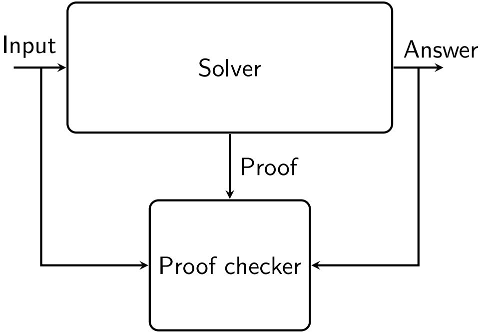
Workflow:

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

CERTIFIED RESULTS WITH PROOF LOGGING

Workflow:

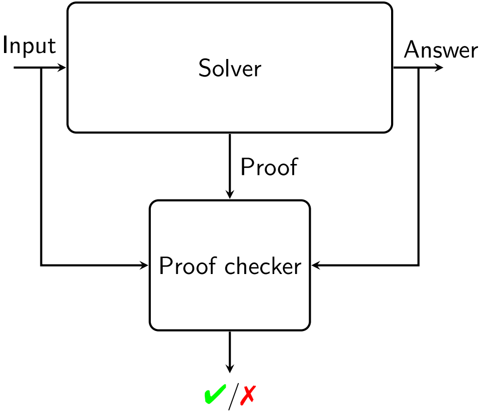
1. Run solver on problem input
2. Get as output not only answer but also proof
3. Feed input + answer + proof to proof checker



CERTIFIED RESULTS WITH PROOF LOGGING

Workflow:

1. Run solver on problem input
2. Get as output not only answer but also proof
3. Feed input + answer + proof to proof checker
4. 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 proof log it
- An introduction to the VeriPB proof system.
- MaxCDCL: Branch-and-Bound with clause learning
- Unweighted MaxCDCL revisited with literal unlocking
- Solution-Improving Constraint using Binary Decision Diagram (BDD) encoding
- Conclusions & Future work

OUTLINE OF THIS PRESENTATION

- MaxSAT and how to proof log it
- An introduction to the VeriPB proof system.
- MaxCDCL: Branch-and-Bound with clause learning
- Unweighted MaxCDCL revisited with literal unlocking
- Solution-Improving Constraint using Binary Decision Diagram (BDD) encoding
- 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:

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

$\mathcal{O} = x_1 + x_2 + x_3$ *(min)*

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:

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

$\mathcal{O} = x_1 + x_2 + x_3$ *(min)*

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:

$F = \{x_1 \vee x_2,\ x_2 \vee x_3,\ x_1 \vee \overline{x_2} \vee x_3\}$
$\mathcal{O} = x_1 + x_2 + x_3$ *(min)*
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 lower 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

Idea (Does not work):

- Utilize one of SAT's proof systems

CERTIFIED MAXSAT SOLVERS

Idea (Does not work):

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

CERTIFIED MAXSAT SOLVERS

Idea (Does not work):

- Utilize one of SAT's proof systems

  Inherently not able to reason about optimality

Idea (Does not work):

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

CERTIFIED MAXSAT SOLVERS

Idea (Does not work):

- Utilize one of SAT's proof systems

  Inherently not able to reason about optimality

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

Idea (Does not work):

- Utilize one of SAT's proof systems

  Inherently not able to reason about optimality

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 $F' = F \wedge \qquad \mathcal{O} < v^*$

CERTIFIED MAXSAT SOLVERS

Idea (Does not work):

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

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 $F' = F \wedge \qquad \mathcal{O} < v^*$
- Run SAT solver with standard proof logging to obtain certificate of UNSAT for F'

CERTIFIED MAXSAT SOLVERS

Idea (Does not work):

- Utilize one of SAT's proof systems

  Inherently not able to reason about optimality

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

Idea (Does not work):

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

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 $F' = F \wedge \text{CNF}(\mathcal{O} < v^*)$
  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

Idea (Does not work):

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

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

Idea (Does not work):

- Utilize one of SAT's proof systems

  Inherently not able to reason about optimality

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

Idea:

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

CERTIFIED MAXSAT SOLVERS

Idea:

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

CERTIFIED MAXSAT SOLVERS

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

Idea:

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

A small and recent history of VeriPB MaxSAT proof logging:

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

CERTIFIED MAXSAT SOLVERS

Idea:

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

A small and recent history of VeriPB MaxSAT proof logging:

- Solution Improving Search
  - QMaxSAT: Focus on proof logging PB-to-CNF encodings [Van23, VDB22]
- Core-Guided Search
  - RC2 and CGSS: First state-of-the-art MaxSAT solver with proof logging [BBN+23]
  - Including techniques such as stratification, hardening, intrinsic-at-most-ones constraints,

CERTIFIED MAXSAT SOLVERS

Idea:

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

A small and recent history of VeriPB MaxSAT proof logging:

- Solution Improving Search
  - QMaxSAT: Focus on proof logging PB-to-CNF encodings [Van23, VDB22]
- Core-Guided Search
  - RC2 and CGSS: First state-of-the-art MaxSAT solver with proof logging [BBN+23]
  - Including techniques such as stratification, hardening, intrinsic-at-most-ones constraints,
- Solution Improving Search revisited
  - Pacose: Challenging without-loss-of-generality reasoning in Dynamic Polynomial Watchdog Encoding [BBN+24]

CERTIFIED MAXSAT SOLVERS

Idea:

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

A small and recent history of VeriPB MaxSAT proof logging:

- Solution Improving Search
  - QMaxSAT: Focus on proof logging PB-to-CNF encodings [Van23, VDB22]
- Core-Guided Search
  - RC2 and CGSS: First state-of-the-art MaxSAT solver with proof logging [BBN+23]
  - Including techniques such as stratification, hardening, intrinsic-at-most-ones constraints,
- Solution Improving Search revisited
  - Pacose: Challenging without-loss-of-generality reasoning in Dynamic Polynomial Watchdog Encoding [BBN+24]
- Branch-and-Bound with clause learning
  - MaxCDCL – **This talk** (and a little bit about *Solution Improving Search*)

OUTLINE OF THIS PRESENTATION

- MaxSAT and how to proof log it
- An introduction to the VeriPB proof system.
- MaxCDCL: Branch-and-Bound with clause learning
- Unweighted MaxCDCL revisited with literal unlocking
- Solution-Improving Constraint using Binary Decision Diagram (BDD) encoding
- 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 $x_i + \overline{x}_i = 1$)

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:

- 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:

- Cutting Planes proof system [CCT87]
  - e.g., adding up two constraints
- Reverse Unit Propagation [GN03]
  - allows deriving constraints without providing an explicit derivation
  - if $F \wedge \neg C \vdash_{\mathsf{UP}} \bot$, then $F \models C$

REASONING OVER PSEUDO-BOOLEAN CONSTRAINTS USING *VeriPB*

VeriPB reasons on such pseudo-Boolean constraints with:

- Cutting Planes proof system [CCT87]
  - e.g., adding up two constraints
- Reverse Unit Propagation [GN03]
  - allows deriving constraints without providing an explicit derivation
  - if $F \wedge \neg C \vdash_{\mathsf{UP}} \bot$, then $F \models C$
- Redundance-Based Strenghtening [GN21, BGMN22]
  - generalisation of the RAT-rule [BT19]
  - allows for proving without-loss-of-generality reasoning
  - e.g. 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:

- Cutting Planes proof system [CCT87]
  - e.g., adding up two constraints
- Reverse Unit Propagation [GN03]
  - allows deriving constraints without providing an explicit derivation
  - if $F \wedge \neg C \vdash_{\text{UP}} \bot$, then $F \models C$
- Redundance-Based Strenghtening [GN21, BGMN22]
  - generalisation of the RAT-rule [BT19]
  - allows for proving without-loss-of-generality reasoning
  - e.g. introducing "fresh" reification variables, such as $r \Leftrightarrow (\sum_i a_i l_i \geq A)$
- Support for Optimisation [BGMN22]
  - allows deriving solution-improving constraints ($\mathcal{O} < v^*$)
  - proving optimality by contradiction

PSEUDO-BOOLEAN REASONING: CUTTING PLANES [CCT87]

Input/model axioms

From the input

PSEUDO-BOOLEAN REASONING: CUTTING PLANES [CCT87]

Input/model axioms From the input

Literal axioms

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

PSEUDO-BOOLEAN REASONING: CUTTING PLANES [CCT87]

Input/model axioms From the input

Literal axioms

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

Addition

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

PSEUDO-BOOLEAN REASONING: CUTTING PLANES [CCT87]

Input/model axioms From the input

Literal axioms

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

Addition

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

Multiplication for any $c \in \mathbb{N}^+$

$$\frac{\sum_i a_i \ell_i \geq A}{\sum_i c a_i \ell_i \geq cA}$$

PSEUDO-BOOLEAN REASONING: CUTTING PLANES [CCT87]

Input/model axioms From the input

Literal axioms

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

Addition

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

Multiplication for any $c \in \mathbb{N}^+$

$$\frac{\sum_i a_i \ell_i \geq A}{\sum_i c a_i \ell_i \geq cA}$$

Division for any $c \in \mathbb{N}^+$
(assumes normalized form)

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

CUTTING PLANES TOY EXAMPLE

$$w + 2x + y \geq 2$$

CUTTING PLANES TOY EXAMPLE

Multiply by 2 $\dfrac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4}$

CUTTING PLANES TOY EXAMPLE

Multiply by 2 $\dfrac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4}$ $\quad w + 2x + 4y + 2z \geq 5$

CUTTING PLANES TOY EXAMPLE

$$\text{Multiply by } 2 \; \dfrac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4}$$

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

CUTTING PLANES TOY EXAMPLE

$$\text{Add}\ \dfrac{\text{Multiply by 2}\ \dfrac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4} \qquad w + 2x + 4y + 2z \geq 5}{3w + 6x + 6y + 2z \geq 9} \qquad \overline{z} \geq 0$$

CUTTING PLANES TOY EXAMPLE

Multiply by 2

$$\frac{\dfrac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4} \qquad w + 2x + 4y + 2z \geq 5}{3w + 6x + 6y + 2z \geq 9}\ \text{Add} \qquad \frac{\overline{z} \geq 0}{2\overline{z} \geq 0}\ \text{Multiply by } 2$$

CUTTING PLANES TOY EXAMPLE

$$\text{Multiply by } 2 \; \dfrac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4} \qquad w + 2x + 4y + 2z \geq 5$$

$$\text{Add} \; \dfrac{2w + 4x + 2y \geq 4 \qquad w + 2x + 4y + 2z \geq 5}{3w + 6x + 6y + 2z \geq 9} \qquad \dfrac{\overline{z} \geq 0}{2\overline{z} \geq 0} \; \text{Multiply by } 2$$

$$\text{Add} \; \dfrac{3w + 6x + 6y + 2z \geq 9 \qquad 2\overline{z} \geq 0}{3w + 6x + 6y + 2z + 2\overline{z} \geq 9}$$

CUTTING PLANES TOY EXAMPLE

Multiply by 2
$$\frac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4}$$

$$w + 2x + 4y + 2z \geq 5$$

Add
$$3w + 6x + 6y + 2z \geq 9$$

Multiply by 2
$$\frac{\overline{z} \geq 0}{2\overline{z} \geq 0}$$

Add
$$3w + 6x + 6y + 2 \qquad \geq 9$$

CUTTING PLANES TOY EXAMPLE

Multiply by 2 $\dfrac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4}$

Add: $\dfrac{2w + 4x + 2y \geq 4 \qquad w + 2x + 4y + 2z \geq 5}{3w + 6x + 6y + 2z \geq 9}$

$\dfrac{\overline{z} \geq 0}{2\overline{z} \geq 0}$ Multiply by 2

Add: $\dfrac{3w + 6x + 6y + 2z \geq 9 \qquad 2\overline{z} \geq 0}{3w + 6x + 6y \qquad\qquad \geq 7}$

CUTTING PLANES TOY EXAMPLE

Multiply by 2: from $w + 2x + y \geq 2$ derive $2w + 4x + 2y \geq 4$

Add: $2w + 4x + 2y \geq 4$ and $w + 2x + 4y + 2z \geq 5$ gives $3w + 6x + 6y + 2z \geq 9$

Multiply by 2: from $\overline{z} \geq 0$ derive $2\overline{z} \geq 0$

Add: $3w + 6x + 6y + 2z \geq 9$ and $2\overline{z} \geq 0$ gives $3w + 6x + 6y \geq 7$

Divide by 3: from $3w + 6x + 6y \geq 7$ derive $w + 2x + 2y \geq 2\frac{1}{3}$

CUTTING PLANES TOY EXAMPLE

Multiply by 2:
$$\frac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4}$$

Multiply by 2:
$$\frac{\overline{z} \geq 0}{2\overline{z} \geq 0}$$

Add:
$$\frac{2w + 4x + 2y \geq 4 \qquad w + 2x + 4y + 2z \geq 5}{3w + 6x + 6y + 2z \geq 9}$$

Add:
$$\frac{3w + 6x + 6y + 2z \geq 9 \qquad 2\overline{z} \geq 0}{3w + 6x + 6y \geq 7}$$

Divide by 3:
$$\frac{3w + 6x + 6y \geq 7}{w + 2x + 2y \geq 3}$$

CUTTING PLANES TOY EXAMPLE

Multiply by 2:
$$\frac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4}$$

Add $w + 2x + 4y + 2z \geq 5$:
$$\frac{2w + 4x + 2y \geq 4 \qquad w + 2x + 4y + 2z \geq 5}{3w + 6x + 6y + 2z \geq 9}$$

Multiply by 2:
$$\frac{\overline{z} \geq 0}{2\overline{z} \geq 0}$$

Add:
$$\frac{3w + 6x + 6y + 2z \geq 9 \qquad 2\overline{z} \geq 0}{3w + 6x + 6y \geq 7}$$

Divide by 3:
$$\frac{3w + 6x + 6y \geq 7}{w + 2x + 2y \geq 3}$$

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

$$\begin{aligned} \text{Constraint 1} &\doteq 2x + y + w \geq 2 \\ \text{Constraint 2} &\doteq 2x + 4y + 2z + w \geq 5 \\ \sim\text{z} &\doteq \overline{z} \geq 0 \end{aligned}$$

CUTTING PLANES TOY EXAMPLE

Multiply by 2: $\dfrac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4}$

Add $2w + 4x + 2y \geq 4$ and $w + 2x + 4y + 2z \geq 5$: $3w + 6x + 6y + 2z \geq 9$

Multiply by 2: $\dfrac{\overline{z} \geq 0}{2\overline{z} \geq 0}$

Add $3w + 6x + 6y + 2z \geq 9$ and $2\overline{z} \geq 0$: $3w + 6x + 6y \geq 7$

Divide by 3: $\dfrac{3w + 6x + 6y \geq 7}{w + 2x + 2y \geq 3}$

Naming constraints by integers and literal axioms by the literal involved (with ~ for negation) as

$$\text{Constraint 1} \doteq 2x + y + w \geq 2$$
$$\text{Constraint 2} \doteq 2x + 4y + 2z + w \geq 5$$
$$\sim\text{z} \doteq \overline{z} \geq 0$$

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

```
p 1 2 * 2 + ~z 2 * + 3 d
```

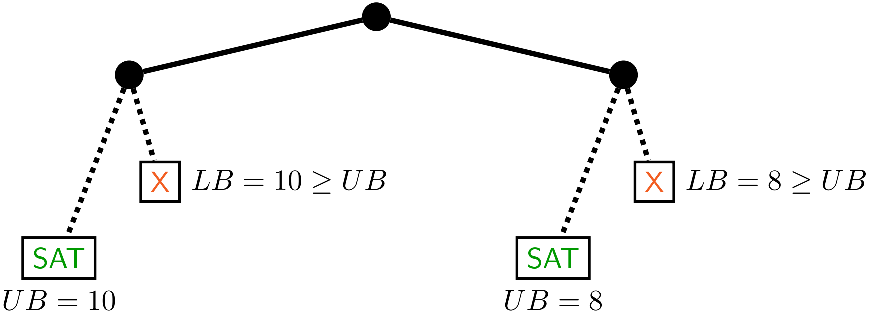
OUTLINE OF THIS PRESENTATION

- MaxSAT and how to proof log it
- An introduction to the VeriPB proof system.
- MaxCDCL: Branch-and-Bound with clause learning
- Unweighted MaxCDCL revisited with literal unlocking
- Solution-Improving Constraint using Binary Decision Diagram (BDD) encoding
- Conclusions & Future work

BRANCH AND BOUND

Branch and Bound:

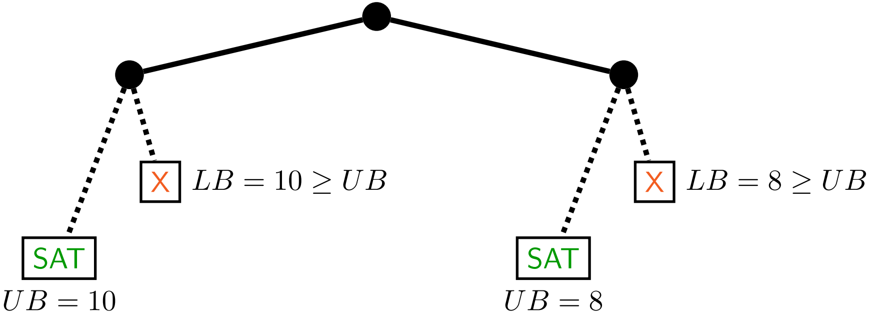
- Explore the search tree for solutions
- Update Upper Bound UB when solution with better objective value is found
- Underestimate Lower Bound LB at every node
- Prune branch when conflict found or when $LB \geq UB$



MAXCDCL AS BRANCH AND BOUND

Branch and Bound in MaxCDCL:

- Explore the search tree for solutions
- Update Upper Bound UB when solution with better objective value is found
- Underestimate Lower Bound LB at every node **using lookahead with UP**
- Prune branch when conflict found or when $LB \geq UB$ **and learn a clause**



MAXCDCL AS CDCL GENERALIZATION

MaxCDCL conflicts:

- **Hard conflict:**
  - A clause is falsified
- **Soft conflict:**
  - (underestimated) LB $\geq$ UB

MAXCDCL AS CDCL GENERALIZATION

MaxCDCL conflicts:

- **Hard conflict:**
  - A clause is falsified
- **Soft conflict:**
  - (underestimated) LB $\geq$ UB

In both cases: conflict analysis for learning new clause (CDCL)

LOOKAHEAD: LB UNDERESTIMATION (UNWEIGHTED CASE)

Lookahead with UP for underestimating LB:

1. Assume unassigned objective literals false and apply UP until:
   - A hard clause is falsified
   - Or a not yet assigned objective literal is assigned 1
2. We have found a **local** unsatisfiable core
3. Since unweighted case: Each **disjoint** core increases the LB by 1
4. When $LB \geq UB$, a soft conflict is found

SOFT CONFLICT DETECTION BY EXAMPLE (UNWEIGHTED CASE)

$\mathcal{O}^t = y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 \quad \mathbf{UB} = 3$

Trail: $x_1^d \; \overline{x_2}^p \; x_3^p \; \overline{x_4}^d \; x_5^p \; x_6^p \; x_7^p$

SOFT CONFLICT DETECTION BY EXAMPLE (UNWEIGHTED CASE)

$\mathcal{O}^t = \cancel{y_1} + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8$ $\quad \mathbf{UB} = 3$

Trail: $x_1^d \; \overline{x_2}^p \; x_3^p \; \overline{x_4}^d \; x_5^p \; x_6^p \; x_7^p$

Find one core:

$x_1^d \; \overline{x_2}^p \; x_3^p \; \overline{x_4}^d \; x_5^p \; x_6^p \; x_7^p \; \overline{y}_1^a \; x_9^p \; x_{10}^p$

SOFT CONFLICT DETECTION BY EXAMPLE (UNWEIGHTED CASE)

$\mathcal{O}^t = \cancel{y_1} + \cancel{y_2} + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 \qquad \mathbf{UB} = 3$

Trail: $x_1^d \; \overline{x_2}^p \; x_3^p \; \overline{x_4}^d \; x_5^p \; x_6^p \; x_7^p$

Find one core:

$x_1^d \; \overline{x_2}^p \; x_3^p \; \overline{x_4}^d \; x_5^p \; x_6^p \; x_7^p \; \overline{y}_1^a \; x_9^p \; x_{10}^p \; \overline{y}_2^a \; \overline{x_{11}}^p$

SOFT CONFLICT DETECTION BY EXAMPLE (UNWEIGHTED CASE)

$\mathcal{O}^t = \cancel{y_1} + \cancel{y_2} + \cancel{y_3} + y_4 + y_5 + y_6 + y_7 + y_8 \qquad \mathbf{UB} = 3$

Trail: $x_1^d \; \overline{x_2}^p \; x_3^p \; \overline{x_4}^d \; x_5^p \; x_6^p \; x_7^p$

Find one core:

$x_1^d \; \overline{x_2}^p \; x_3^p \; \overline{x_4}^d \; x_5^p \; x_6^p \; x_7^p \; \overline{y}_1^a \; x_9^p \; x_{10}^p \; \overline{y}_2^a \; \overline{x_{11}}^p \; \overline{y}_3^a$

SOFT CONFLICT DETECTION BY EXAMPLE (UNWEIGHTED CASE)

$\mathcal{O}^t = \cancel{y_1} + \cancel{y_2} + \cancel{y_3} + \cancel{y_4} + y_5 + y_6 + y_7 + y_8 \quad \mathbf{UB} = 3$

Trail: $x_1^d \ \overline{x_2}^p \ x_3^p \ \overline{x_4}^d \ x_5^p \ x_6^p \ x_7^p$

Find one core:

$x_1^d \ \overline{x_2}^p \ x_3^p \ \overline{x_4}^d \ x_5^p \ x_6^p \ x_7^p \ \overline{y}_1^a \ x_9^p \ x_{10}^p \ \overline{y}_2^a \ \overline{x_{11}}^p \ \overline{y}_3^a \ \overline{y}_4^a \ x_{12}^p$ ($\overline{x_{12}} \vee x_{11} \in F$ falsified)

SOFT CONFLICT DETECTION BY EXAMPLE (UNWEIGHTED CASE)

$\mathcal{O}^t = \cancel{y_1} + \cancel{y_2} + \cancel{y_3} + \cancel{y_4} + y_5 + y_6 + y_7 + y_8 \qquad \mathbf{UB} = 3$

Trail: $x_1^d \; \overline{x_2}^p \; x_3^p \; \overline{x_4}^d \; x_5^p \; x_6^p \; x_7^p$

Find one core:

$x_1^d \; \overline{x_2}^p \; x_3^p \; \overline{x_4}^d \; x_5^p \; x_6^p \; x_7^p \; \overline{y_1}^a \; x_9^p \; x_{10}^p \; \overline{y_2}^a \; \overline{x_{11}}^p \; \overline{y_3}^a \; \overline{y_4}^a \; x_{12}^p$ ($\overline{x_{12}} \vee x_{11} \in F$ falsified)

$x_1^d \; \overline{x_2}^p \; x_3^p \; \overline{x_4}^d \; x_5^p \; x_6^p \; x_7^p \; \overline{y_1}^a \; \overline{y_2}^a \; \overline{y_3}^a \; \overline{y_4}^a$ (Assumptions suffice)

SOFT CONFLICT DETECTION BY EXAMPLE (UNWEIGHTED CASE)

$\mathcal{O}^t = \cancel{y_1} + y_2 + y_3 + \cancel{y_4} + y_5 + y_6 + y_7 + y_8 \qquad \mathbf{UB} = 3$

Trail: $x_1^d\ \overline{x_2}^p\ x_3^p\ \overline{x_4}^d\ x_5^p\ x_6^p\ x_7^p$

Find one core:

$x_1^d\ \overline{x_2}^p\ x_3^p\ \overline{x_4}^d\ x_5^p\ x_6^p\ x_7^p\ \overline{y_1}^a\ x_9^p\ x_{10}^p\ \overline{y_2}^a\ \overline{x_{11}}^p\ \overline{y_3}^a\ \overline{y_4}^a\ x_{12}^p$ ($\overline{x_{12}} \vee x_{11} \in F$ falsified)

$x_1^d\ \overline{x_2}^p\ x_3^p\ \overline{x_4}^d\ x_5^p\ x_6^p\ x_7^p\ \overline{y_1}^a\ \overline{y_2}^a\ \overline{y_3}^a\ \overline{y_4}^a$ (Assumptions suffice)

$\overline{x_2}^p\ \overline{x_4}^d\ \overline{y_1}^a\ \overline{y_4}^a$ (Conflict analysis)

SOFT CONFLICT DETECTION BY EXAMPLE (UNWEIGHTED CASE)

$\mathcal{O}^t = \cancel{y_1} + y_2 + y_3 + \cancel{y_4} + y_5 + y_6 + y_7 + y_8 \qquad \mathbf{UB} = 3$

Trail: $x_1^d\ \overline{x_2}^p\ x_3^p\ \overline{x_4}^d\ x_5^p\ x_6^p\ x_7^p$

Find one core:

$x_1^d\ \overline{x_2}^p\ x_3^p\ \overline{x_4}^d\ x_5^p\ x_6^p\ x_7^p\ \overline{y}_1^a\ x_9^p\ x_{10}^p\ \overline{y}_2^a\ \overline{x_{11}}^p\ \overline{y}_3^a\ \overline{y}_4^a\ x_{12}^p$ ($\overline{x_{12}} \vee x_{11} \in F$ falsified)

$x_1^d\ \overline{x_2}^p\ x_3^p\ \overline{x_4}^d\ x_5^p\ x_6^p\ x_7^p\ \overline{y}_1^a\ \overline{y}_2^a\ \overline{y}_3^a\ \overline{y}_4^a$ (Assumptions suffice)

$\overline{x_2}^p\ \overline{x_4}^d\ \overline{y}_1^a\ \overline{y}_4^a$ (Conflict analysis)

Local core:

$\overline{x_2} \wedge \overline{x_4} \wedge \overline{y}_1 \wedge \overline{y}_4 \vdash_{\mathsf{UP}} \square$

$\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_4$ (Reasons $\rightarrow$ Core)

SOFT CONFLICT DETECTION BY EXAMPLE (UNWEIGHTED CASE)

$\mathcal{O} = y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 \qquad \mathbf{UB} = 3$

Trail: $x_1^d\ \overline{x_2}^p\ x_3^p\ \overline{x_4}^d\ x_5^p\ x_6^p\ x_7^p$

Found disjoint local cores

Core 1: $\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_4$

Core 2: $\overline{x_2} \wedge x_7 \rightarrow y_2 \vee y_3 \vee y_5$

Core 3: $x_1 \wedge \overline{x_4} \wedge x_7 \rightarrow y_6 \vee y_7$

SOFT CONFLICT DETECTION BY EXAMPLE (UNWEIGHTED CASE)

$\mathcal{O} = y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 \qquad \mathbf{UB} = 3$

Trail: $x_1^d \; \overline{x_2}^p \; x_3^p \; \overline{x_4}^d \; x_5^p \; x_6^p \; x_7^p$

Found disjoint local cores

Core 1: $\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_4$

Core 2: $\overline{x_2} \wedge x_7 \rightarrow y_2 \vee y_3 \vee y_5$

Core 3: $x_1 \wedge \overline{x_4} \wedge x_7 \rightarrow y_6 \vee y_7$

$x_1 \wedge \overline{x_2} \wedge \overline{x_4} \wedge x_7 \rightarrow (y_1 \vee y_4) \wedge (y_2 \vee y_3 \vee y_5) \wedge (y_6 \vee y_7)$

SOFT CONFLICT DETECTION BY EXAMPLE (UNWEIGHTED CASE)

$\mathcal{O} = y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8$ $\quad \mathbf{UB} = 3$

Trail: $x_1^d \; \overline{x_2}^p \; x_3^p \; \overline{x_4}^d \; x_5^p \; x_6^p \; x_7^p$

Found disjoint local cores

Core 1: $\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_4$

Core 2: $\overline{x_2} \wedge x_7 \rightarrow y_2 \vee y_3 \vee y_5$

Core 3: $x_1 \wedge \overline{x_4} \wedge x_7 \rightarrow y_6 \vee y_7$

$x_1 \wedge \overline{x_2} \wedge \overline{x_4} \wedge x_7 \rightarrow (y_1 \vee y_4) \wedge (y_2 \vee y_3 \vee y_5) \wedge (y_6 \vee y_7)$

$x_1 \wedge \overline{x_2} \wedge \overline{x_4} \wedge x_7 \rightarrow LB = 3 \geq 3 = UB$

SOFT CONFLICT DETECTION BY EXAMPLE (UNWEIGHTED CASE)

$\mathcal{O} = y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8$ $\quad \mathbf{UB} = 3$

Trail: $x_1^d\ \overline{x_2}^p\ x_3^p\ \overline{x_4}^d\ x_5^p\ x_6^p\ x_7^p$

Found disjoint local cores

Core 1: $\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_4$

Core 2: $\overline{x_2} \wedge x_7 \rightarrow y_2 \vee y_3 \vee y_5$

Core 3: $x_1 \wedge \overline{x_4} \wedge x_7 \rightarrow y_6 \vee y_7$

$x_1 \wedge \overline{x_2} \wedge \overline{x_4} \wedge x_7 \rightarrow (y_1 \vee y_4) \wedge (y_2 \vee y_3 \vee y_5) \wedge (y_6 \vee y_7)$

$x_1 \wedge \overline{x_2} \wedge \overline{x_4} \wedge x_7 \rightarrow LB = 3 \geq 3 = UB$

Soft conflict

Conflicting clause: $\overline{x_1} \vee x_2 \vee x_4 \vee \overline{x_7}$

SOFT CONFLICT DETECTION (WEIGHTED CASE)

Weighted MaxCDCL

- Weight of Local Core $\mathcal{K}$ = smallest coefficient of objective literals in $\mathcal{K}$
- Each objective literal can contribute to many cores
- The total contribution of an objective literal cannot exceed its coefficient

SOFT CONFLICT DETECTION (WEIGHTED CASE)

Weighted MaxCDCL

- Weight of Local Core $\mathcal{K}$ = smallest coefficient of objective literals in $\mathcal{K}$
- Each objective literal can contribute to many cores
- The total contribution of an objective literal cannot exceed its coefficient

$\mathcal{O}^t = 7y_1 + 2y_2 + 1y_3 + 1y_4 + 1y_5 + 4y_6 + 1y_7 + 3y_8 \quad \mathbf{UB} = 4$

Trail: $x_1^d\ \overline{x_2}^p\ x_3^p\ \overline{x_4}^d\ x_5^p\ x_6^p\ x_7^p$

Found local cores

SOFT CONFLICT DETECTION (WEIGHTED CASE)

Weighted MaxCDCL

- Weight of Local Core $\mathcal{K}$ = smallest coefficient of objective literals in $\mathcal{K}$
- Each objective literal can contribute to many cores
- The total contribution of an objective literal cannot exceed its coefficient

$\mathcal{O}^t = 7y_1 + 2y_2 + 1y_3 + 1y_4 + 1y_5 + 4y_6 + 1y_7 + 3y_8 \quad \mathbf{UB} = 4$

Trail: $x_1^d\ \overline{x_2}^p\ x_3^p\ \overline{x_4}^d\ x_5^p\ x_6^p\ x_7^p$

Found local cores

Core 1: $\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_2$ (weight 2)

SOFT CONFLICT DETECTION (WEIGHTED CASE)

Weighted MaxCDCL

- Weight of Local Core $\mathcal{K}$ = smallest coefficient of objective literals in $\mathcal{K}$
- Each objective literal can contribute to many cores
- The total contribution of an objective literal cannot exceed its coefficient

$\mathcal{O}^t = \not{7}\ 5y_1 + \not{2}\ 0y_2 + 1y_3 + 1y_4 + 1y_5 + 4y_6 + 1y_7 + 3y_8 \quad \mathbf{UB} = 4$

Trail: $x_1^d\ \overline{x_2}^p\ x_3^p\ \overline{x_4}^d\ x_5^p\ x_6^p\ x_7^p$

Found local cores

Core 1: $\overline{x_2} \land \overline{x_4} \rightarrow y_1 \lor y_2$ (weight 2)

SOFT CONFLICT DETECTION (WEIGHTED CASE)

Weighted MaxCDCL

- Weight of Local Core $\mathcal{K}$ = smallest coefficient of objective literals in $\mathcal{K}$
- Each objective literal can contribute to many cores
- The total contribution of an objective literal cannot exceed its coefficient

$\mathcal{O}^t = \not{7}\ 5y_1 + \not{2}\ 0y_2 + 1y_3 + 1y_4 + 1y_5 + 4y_6 + 1y_7 + 3y_8 \quad \textbf{UB} = 4$

Trail: $x_1^d\ \overline{x_2}^p\ x_3^p\ \overline{x_4}^d\ x_5^p\ x_6^p\ x_7^p$

Found local cores

Core 1: $\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_2$ (weight 2)

Core 2: $x_3 \wedge \overline{x_4} \rightarrow y_1 \vee y_5$ (weight 1)

SOFT CONFLICT DETECTION (WEIGHTED CASE)

Weighted MaxCDCL

- Weight of Local Core $\mathcal{K}$ = smallest coefficient of objective literals in $\mathcal{K}$
- Each objective literal can contribute to many cores
- The total contribution of an objective literal cannot exceed its coefficient

$\mathcal{O}^t = \not{7}\ \not{5}\ 4y_1 + \not{2}\ 0y_2 + 1y_3 + 1y_4 + \not{1}\ 0y_5 + 4y_6 + 1y_7 + 3y_8 \quad \mathbf{UB} = 4$

Trail: $x_1^d\ \overline{x_2}^p\ x_3^p\ \overline{x_4}^d\ x_5^p\ x_6^p\ x_7^p$

Found local cores

Core 1: $\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_2$ (weight 2)

Core 2: $x_3 \wedge \overline{x_4} \rightarrow y_1 \vee y_5$ (weight 1)

SOFT CONFLICT DETECTION (WEIGHTED CASE)

Weighted MaxCDCL

- Weight of Local Core $\mathcal{K}$ = smallest coefficient of objective literals in $\mathcal{K}$
- Each objective literal can contribute to many cores
- The total contribution of an objective literal cannot exceed its coefficient

$\mathcal{O}^t = \not{7}\ \not{5}\ \not{4}\ 1y_1 + \not{2}\ 0y_2 + 1y_3 + 1y_4 + \not{1}\ 0y_5 + \not{4}\ 1y_6 + 1y_7 + \not{3}\ 0y_8 \quad \mathbf{UB} = 4$

Trail: $x_1^d\ \overline{x_2}^p\ x_3^p\ \overline{x_4}^d\ x_5^p\ x_6^p\ x_7^p$

Found local cores

Core 1: $\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_2$ (weight 2)

Core 2: $x_3 \wedge \overline{x_4} \rightarrow y_1 \vee y_5$ (weight 1)

Core 3: $x_1 \rightarrow y_1 \vee y_6 \vee y_8$ (weight 3)

SOFT CONFLICT DETECTION (WEIGHTED CASE)

Weighted MaxCDCL

- Weight of Local Core $\mathcal{K}$ = smallest coefficient of objective literals in $\mathcal{K}$
- Each objective literal can contribute to many cores
- The total contribution of an objective literal cannot exceed its coefficient

$\mathcal{O}^t = \cancel{7}\ \cancel{5}\ 2y_1 + \cancel{2}\ 0y_2 + 1y_3 + 1y_4 + 1y_5 + \cancel{4}\ 1y_6 + 1y_7 + \cancel{3}\ 0y_8 \qquad \textbf{UB} = 4$

Trail: $x_1^d\ \overline{x_2}^p\ x_3^p\ \overline{x_4}^d\ x_5^p\ x_6^p\ x_7^p$

Found local cores

Core 1: $\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_2$ (weight 2)

~~Core 2: $x_3 \wedge \overline{x_4} \rightarrow y_1 \vee y_5$ (weight 1)~~

Core 3: $x_1 \rightarrow y_1 \vee y_6 \vee y_8$ (weight 3)

SOFT CONFLICT DETECTION (WEIGHTED CASE)

Weighted MaxCDCL

- Weight of Local Core $\mathcal{K}$ = smallest coefficient of objective literals in $\mathcal{K}$
- Each objective literal can contribute to many cores
- The total contribution of an objective literal cannot exceed its coefficient

$\mathcal{O}^t = \not{7}\ \not{5}\ 2y_1 + \not{2}\ 0y_2 + 1y_3 + 1y_4 + 1y_5 + \not{4}\ 1y_6 + 1y_7 + \not{3}\ 0y_8 \quad \mathbf{UB} = 4$

Trail: $x_1^d\ \overline{x_2}^p\ x_3^p\ \overline{x_4}^d\ x_5^p\ x_6^p\ x_7^p$

Found local cores

Core 1: $\overline{x_2} \land \overline{x_4} \rightarrow y_1 \lor y_2$ (weight 2)

Core 3: $x_1 \rightarrow y_1 \lor y_6 \lor y_8$ (weight 3)

SOFT CONFLICT DETECTION (WEIGHTED CASE)

Weighted MaxCDCL

- Weight of Local Core $\mathcal{K}$ = smallest coefficient of objective literals in $\mathcal{K}$
- Each objective literal can contribute to many cores
- The total contribution of an objective literal cannot exceed its coefficient

$\mathcal{O}^t = \cancel{7}\ \cancel{5}\ 2y_1 + \cancel{2}\ 0y_2 + 1y_3 + 1y_4 + 1y_5 + \cancel{4}\ 1y_6 + 1y_7 + \cancel{3}\ 0y_8$ $\quad$ $\mathbf{UB} = 4$

Trail: $x_1^d\ \overline{x_2}^p\ x_3^p\ \overline{x_4}^d\ x_5^p\ x_6^p\ x_7^p$

Found local cores

Core 1: $\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_2$ (weight 2)

Core 3: $x_1 \rightarrow y_1 \vee y_6 \vee y_8$ (weight 3)

Conclusion $x_1 \wedge \overline{x_2} \wedge \overline{x_4} \rightarrow LB = 5 \geq 4 = UB$

SOFT CONFLICT DETECTION (WEIGHTED CASE)

Weighted MaxCDCL

- Weight of Local Core $\mathcal{K}$ = smallest coefficient of objective literals in $\mathcal{K}$
- Each objective literal can contribute to many cores
- The total contribution of an objective literal cannot exceed its coefficient

$\mathcal{O}^t = \not 7\ \not 5\ 2y_1 + \not 2\ 0y_2 + 1y_3 + 1y_4 + 1y_5 + \not 4\ 1y_6 + 1y_7 + \not 3\ 0y_8 \qquad \mathbf{UB} = 4$

Trail: $x_1^d\ \overline{x_2}^p\ x_3^p\ \overline{x_4}^d\ x_5^p\ x_6^p\ x_7^p$

Found local cores

Core 1: $\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_2$ (weight 2)

Core 3: $x_1 \rightarrow y_1 \vee y_6 \vee y_8$ (weight 3)

Conclusion $x_1 \wedge \overline{x_2} \wedge \overline{x_4} \rightarrow LB = 5 \geq 4 = UB$

Soft conflict Conflicting clause: $\overline{x}_1 \vee x_2 \vee x_4$

PROOF LOGGING SOFT CONFLICTS

To Derive: $\overline{x}_1 + x_2 + x_4 \geq 1 \quad \mathbf{UB} = 4$

PROOF LOGGING SOFT CONFLICTS

To Derive: $\overline{x}_1 + x_2 + x_4 \geq 1$ $\quad \mathbf{UB} = 4$

Found "disjoint" cores

Core 1: $\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_2$ (2)

Core 2: $x_1 \rightarrow y_1 \vee y_6 \vee y_8$ (3)

PROOF LOGGING SOFT CONFLICTS

To Derive: $\overline{x}_1 + x_2 + x_4 \geq 1$ $\quad \mathbf{UB} = 4$

Found "disjoint" cores

Core 1: $\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_2$ (2)

PB: $x_2 + x_4 + y_1 + y_2 \geq 1$

Core 2: $x_1 \rightarrow y_1 \vee y_6 \vee y_8$ (3)

PB: $\overline{x}_1 + y_1 + y_6 + y_8 \geq 1$

PROOF LOGGING SOFT CONFLICTS

To Derive: $\overline{x}_1 + x_2 + x_4 \geq 1$ $\quad \mathbf{UB} = 4$

Found "disjoint" cores (RUP)

Core 1: $\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_2$ (2)

PB: $x_2 + x_4 + y_1 + y_2 \geq 1$

Core 2: $x_1 \rightarrow y_1 \vee y_6 \vee y_8$ (3)

PB: $\overline{x}_1 + y_1 + y_6 + y_8 \geq 1$

PROOF LOGGING SOFT CONFLICTS

To Derive: $\overline{x}_1 + x_2 + x_4 \geq 1$ $\quad \mathbf{UB} = 4$

Found "disjoint" cores (RUP)

Core 1: $\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_2$ (2)

PB: $2x_2 + 2x_4 + 2y_1 + 2y_2 \geq 2 \cancel{1}$

Core 2: $x_1 \rightarrow y_1 \vee y_6 \vee y_8$ (3)

PB: $3\overline{x}_1 + 3y_1 + 3y_6 + 3y_8 \geq 3 \cancel{1}$

Multiplication by their weight

PROOF LOGGING SOFT CONFLICTS

To Derive: $\overline{x}_1 + x_2 + x_4 \geq 1$ $\quad \mathbf{UB} = 4$

Found "disjoint" cores (RUP)

Core 1: $\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_2$ (2)

PB: $2x_2 + 2x_4 + 2y_1 + 2y_2 \geq 2 \not{1}$

Core 2: $x_1 \rightarrow y_1 \vee y_6 \vee y_8$ (3)

PB: $3\overline{x}_1 + 3y_1 + 3y_6 + 3y_8 \geq 3 \not{1}$

Multiplication by their weight and addition:

$3\overline{x}_1 + 2x_2 + 2x_4 + 5y_1 + 2y_2 + 3y_6 + 3y_8 \geq 5$

PROOF LOGGING SOFT CONFLICTS

To Derive: $\overline{x}_1 + x_2 + x_4 \geq 1$ $\quad \mathbf{UB} = 4$

Found "disjoint" cores (RUP)

Core 1: $\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_2$ (2)

PB: $2x_2 + 2x_4 + 2y_1 + 2y_2 \geq 2 \not{1}$

Core 2: $x_1 \rightarrow y_1 \vee y_6 \vee y_8$ (3)

PB: $3\overline{x}_1 + 3y_1 + 3y_6 + 3y_8 \geq 3 \not{1}$

Multiplication by their weight and addition:

$3\overline{x}_1 + 2x_2 + 2x_4 + 5y_1 + 2y_2 + 3y_6 + 3y_8 \geq 5$

Model improving constraint

$7y_1 + 2y_2 + 1y_3 + 1y_4 + 1y_5 + 4y_6 + 1y_7 + 3y_8 \leq 3$

PROOF LOGGING SOFT CONFLICTS

To Derive: $\overline{x}_1 + x_2 + x_4 \geq 1$ $\mathbf{UB} = 4$

Found "disjoint" cores (RUP)

Core 1: $\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_2$ (2)

PB: $2x_2 + 2x_4 + 2y_1 + 2y_2 \geq 2 \not{1}$

Core 2: $x_1 \rightarrow y_1 \vee y_6 \vee y_8$ (3)

PB: $3\overline{x}_1 + 3y_1 + 3y_6 + 3y_8 \geq 3 \not{1}$

Multiplication by their weight and addition:

$3\overline{x}_1 + 2x_2 + 2x_4 + 5y_1 + 2y_2 + 3y_6 + 3y_8 \geq 5$

Model improving constraint

$7y_1 + 2y_2 + 1y_3 + 1y_4 + 1y_5 + 4y_6 + 1y_7 + 3y_8 \leq 3$

In normalized form:

$7\overline{y}_1 + 2\overline{y}_2 + 1\overline{y}_3 + 1\overline{y}_4 + 1\overline{y}_5 + 4\overline{y}_6 + 1\overline{y}_7 + 3\overline{y}_8 \geq 20 - 3$

PROOF LOGGING SOFT CONFLICTS

To Derive: $\overline{x}_1 + x_2 + x_4 \geq 1$ $\quad \mathbf{UB} = 4$

Found "disjoint" cores (RUP)

Core 1: $\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_2$ (2)

PB: $2x_2 + 2x_4 + 2y_1 + 2y_2 \geq 2$ ~~1~~

Core 2: $x_1 \rightarrow y_1 \vee y_6 \vee y_8$ (3)

PB: $3\overline{x}_1 + 3y_1 + 3y_6 + 3y_8 \geq 3$ ~~1~~

Multiplication by their weight and addition:

$3\overline{x}_1 + 2x_2 + 2x_4 + 5y_1 + 2y_2 + 3y_6 + 3y_8 \geq 5$

Model improving constraint

$7y_1 + 2y_2 + 1y_3 + 1y_4 + 1y_5 + 4y_6 + 1y_7 + 3y_8 \leq 3$

In normalized form:

$7\overline{y}_1 + 2\overline{y}_2 + 1\overline{y}_3 + 1\overline{y}_4 + 1\overline{y}_5 + 4\overline{y}_6 + 1\overline{y}_7 + 3\overline{y}_8 \geq 20 - 3$

By adding literal axioms:

$5\overline{y}_1 + 2\overline{y}_2 + 3\overline{y}_6 + 3\overline{y}_8 \geq 13 - 3$

PROOF LOGGING SOFT CONFLICTS

To Derive: $\overline{x}_1 + x_2 + x_4 \geq 1$ $\quad \mathbf{UB} = 4$

Found "disjoint" cores (RUP)

Core 1: $\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_2$ (2)

PB: $2x_2 + 2x_4 + 2y_1 + 2y_2 \geq 2$ ~~1~~

Core 2: $x_1 \rightarrow y_1 \vee y_6 \vee y_8$ (3)

PB: $3\overline{x}_1 + 3y_1 + 3y_6 + 3y_8 \geq 3$ ~~1~~

Multiplication by their weight and addition:

$3\overline{x}_1 + 2x_2 + 2x_4 + 5y_1 + 2y_2 + 3y_6 + 3y_8 \geq 5$

Model improving constraint

$7y_1 + 2y_2 + 1y_3 + 1y_4 + 1y_5 + 4y_6 + 1y_7 + 3y_8 \leq 3$

In normalized form:

$7\overline{y}_1 + 2\overline{y}_2 + 1\overline{y}_3 + 1\overline{y}_4 + 1\overline{y}_5 + 4\overline{y}_6 + 1\overline{y}_7 + 3\overline{y}_8 \geq 20 - 3$

By adding literal axioms:

$5\overline{y}_1 + 2\overline{y}_2 + 3\overline{y}_6 + 3\overline{y}_8 \geq 13 - 3$

Addition:

$$3\overline{x}_1 + 2x_2 + 2x_4 + 5y_1 + 5\overline{y}_1 + 2y_2 + 2\overline{y}_2 + 3y_6 + 3\overline{y}_6 + 3y_8 + 3\overline{y}_8 \geq 13 + 5 - 3$$

PROOF LOGGING SOFT CONFLICTS

To Derive: $\overline{x}_1 + x_2 + x_4 \geq 1$ $\quad \mathbf{UB} = 4$

Found "disjoint" cores (RUP)

Core 1: $\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_2$ (2)

PB: $2x_2 + 2x_4 + 2y_1 + 2y_2 \geq 2 \not{1}$

Core 2: $x_1 \rightarrow y_1 \vee y_6 \vee y_8$ (3)

PB: $3\overline{x}_1 + 3y_1 + 3y_6 + 3y_8 \geq 3 \not{1}$

Multiplication by their weight and addition:

$3\overline{x}_1 + 2x_2 + 2x_4 + 5y_1 + 2y_2 + 3y_6 + 3y_8 \geq 5$

Model improving constraint

$7y_1 + 2y_2 + 1y_3 + 1y_4 + 1y_5 + 4y_6 + 1y_7 + 3y_8 \leq 3$

In normalized form:

$7\overline{y}_1 + 2\overline{y}_2 + 1\overline{y}_3 + 1\overline{y}_4 + 1\overline{y}_5 + 4\overline{y}_6 + 1\overline{y}_7 + 3\overline{y}_8 \geq 20 - 3$

By adding literal axioms:

$5\overline{y}_1 + 2\overline{y}_2 + 3\overline{y}_6 + 3\overline{y}_8 \geq 13 - 3$

Addition:

$$3\overline{x}_1 + 2x_2 + 2x_4 + \cancel{5y_1 + 5\overline{y}_1 + 2y_2 + 2\overline{y}_2 + 3y_6 + 3\overline{y}_6 + 3y_8 + 3\overline{y}_8} \geq \cancel{13} + 5 - 3$$

PROOF LOGGING SOFT CONFLICTS

To Derive: $\overline{x}_1 + x_2 + x_4 \geq 1 \quad \mathbf{UB} = 4$

Found “disjoint” cores (RUP)

Core 1: $\overline{x_2} \wedge \overline{x_4} \rightarrow y_1 \vee y_2$ (2)

PB: $2x_2 + 2x_4 + 2y_1 + 2y_2 \geq 2 \not{1}$

Core 2: $x_1 \rightarrow y_1 \vee y_6 \vee y_8$ (3)

PB: $3\overline{x}_1 + 3y_1 + 3y_6 + 3y_8 \geq 3 \not{1}$

Multiplication by their weight and addition:

$3\overline{x}_1 + 2x_2 + 2x_4 + 5y_1 + 2y_2 + 3y_6 + 3y_8 \geq 5$

Model improving constraint

$7y_1 + 2y_2 + 1y_3 + 1y_4 + 1y_5 + 4y_6 + 1y_7 + 3y_8 \leq 3$

In normalized form:

$7\overline{y}_1 + 2\overline{y}_2 + 1\overline{y}_3 + 1\overline{y}_4 + 1\overline{y}_5 + 4\overline{y}_6 + 1\overline{y}_7 + 3\overline{y}_8 \geq 20 - 3$

By adding literal axioms:

$5\overline{y}_1 + 2\overline{y}_2 + 3\overline{y}_6 + 3\overline{y}_8 \geq 13 - 3$

Addition:

$3\overline{x}_1 + 2x_2 + 2x_4 + \cancel{5y_1 + 5\overline{y}_1 + 2y_2 + 2\overline{y}_2 + 3y_6 + 3\overline{y}_6 + 3y_8 + 3\overline{y}_8} \geq \cancel{13} + 5 - 3$

Division by a large enough number (and rounding up): $\overline{x}_1 + x_2 + x_4 \geq 1$

PROOF LOGGING MAXCDCL

Proof logging Learned clause after conflict analysis RUP

PROOF LOGGING MAXCDCL

Proof logging Learned clause after conflict analysis RUP

Proof logging Optimality:

- Unit propagation in MaxCDCL derives conflict at DL $= 0$
- Proof: RUP $0 \geq 1$

OUTLINE OF THIS PRESENTATION

- MaxSAT and how to proof log it
- An introduction to the VeriPB proof system.
- MaxCDCL: Branch-and-Bound with clause learning
- Unweighted MaxCDCL revisited with literal unlocking
- Solution-Improving Constraint using Binary Decision Diagram (BDD) encoding
- Conclusions & Future work

UNWEIGHTED MAXCDCL REVISITED

Unweighted MaxCDCL searches for set $\mathcal{L}$ of tuples (b, L) such that

1. Each L is a set of objective literals
2. For each (b, L) in $\mathcal{L}$, it holds that $F \wedge \alpha \models \sum_{\ell \in L} \ell \geq b$.
3. For each pair (b, L) and (b', L') in $\mathcal{L}$, $L \cap L' = \emptyset$.
4. The total weight exceeds the current upper bound: $\sum_{(b,L) \in \mathcal{L}} b \geq \mathbf{UB}$.

UNWEIGHTED MAXCDCL REVISITED

Unweighted MaxCDCL searches for set $\mathcal{L}$ of tuples (b, L) such that

1. Each L is a set of objective literals
2. For each (b, L) in $\mathcal{L}$, it holds that $F \wedge \alpha \models \sum_{\ell \in L} \ell \geq b$.
3. For each pair (b, L) and (b', L') in $\mathcal{L}$, $L \cap L' = \emptyset$.
4. The total weight exceeds the current upper bound: $\sum_{(b,L)\in\mathcal{L}} b \geq \mathbf{UB}$.

$\mathcal{O} = y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 + y_9 + \ldots \qquad \mathbf{UB} = 4$

Found disjoint local "cores"

Core 1: $\overline{x}_2 \wedge \overline{x}_4 \rightarrow y_1 + y_3 + y_5 + y_8 \geq 3$

Core 2: $x_4 \wedge \overline{x}_7 \wedge x_9 \rightarrow y_2 + y_4 + y_6 \geq 2$

UNWEIGHTED MAXCDCL REVISITED

Unweighted MaxCDCL searches for set $\mathcal{L}$ of tuples (b, L) such that

1. Each L is a set of objective literals
2. For each (b, L) in $\mathcal{L}$, it holds that $F \wedge \alpha \models \sum_{\ell \in L} \ell \geq b$.
3. For each pair (b, L) and (b', L') in $\mathcal{L}$, $L \cap L' = \emptyset$.
4. The total weight exceeds the current upper bound: $\sum_{(b,L) \in \mathcal{L}} b \geq \mathbf{UB}$.

$\mathcal{O} = y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 + y_9 + \ldots \quad \mathbf{UB} = 4$

Found disjoint local "cores"

Core 1: $\overline{x}_2 \wedge \overline{x}_4 \rightarrow y_1 + y_3 + y_5 + y_8 \geq 3$

Core 2: $x_4 \wedge \overline{x}_7 \wedge x_9 \rightarrow y_2 + y_4 + y_6 \geq 2$

$\overline{x}_2 \wedge \overline{x}_4 \wedge \overline{x}_7 \wedge x_9 \rightarrow LB = 5 \geq 4 = UB$ **Soft conflict clause:** $x_2 \vee x_4 \vee x_7 \vee \overline{x}_9$

LOOKAHEAD WITH LITERAL UNLOCKING (BY EXAMPLE)

$\mathcal{O}^t = \cancel{y_1} + \cancel{y_2} + \cancel{y_3} + \cancel{y_4} + \cancel{y_5} + \cancel{y_6} + \cancel{y_7} + \cancel{y_8} + y_9 + \ldots$

Trail: $x_1^d\ \overline{x_2}^d\ x_3^p\ \overline{x_4}^d\ x_5^p$

Found disjoint local "cores"

Core 1: $\overline{x_2} \land \overline{x_4} \rightarrow y_3 + y_5 + y_6 \geq 1$

Core 2: $x_1 \land \overline{x_2} \rightarrow y_1 + y_2 + y_4 + y_7 + y_8 \geq 2$

LOOKAHEAD WITH LITERAL UNLOCKING (BY EXAMPLE)

$\mathcal{O}^t = \cancel{y_1} + \cancel{y_2} + \cancel{y_3} + \cancel{y_4} + \cancel{y_5} + \cancel{y_6} + \cancel{y_7} + \cancel{y_8} + y_9 + \ldots$

Trail: $x_1^d \; \overline{x_2}^d \; x_3^p \; \overline{x_4}^d \; x_5^p$

Found disjoint local "cores"

Core 1: $\cancel{\overline{x_2} \wedge \overline{x_4}} \rightarrow y_3 + y_5 + y_6 \geq 1$

Core 2: $\cancel{x_1 \wedge \overline{x_2}} \rightarrow y_1 + y_2 + y_4 + y_7 + y_8 \geq 2$

LOOKAHEAD WITH LITERAL UNLOCKING (BY EXAMPLE)

$\mathcal{O}^t = \cancel{y_1} + \cancel{y_2} + \cancel{y_3} + \cancel{y_4} + \cancel{y_5} + \cancel{y_6} + \cancel{y_7} + \cancel{y_8} + y_9 + \ldots$

Trail: $x_1^d \; \overline{x_2}^d \; x_3^p \; \overline{x_4}^d \; x_5^p$

Found disjoint local "cores"

Core 1: $y_3 + y_5 + y_6 \geq 1$

Core 2: $y_1 + y_2 + y_4 + y_7 + y_8 \geq 2$

LOOKAHEAD WITH LITERAL UNLOCKING (BY EXAMPLE)

$\mathcal{O}^t = \cancel{y_1} + \cancel{y_2} + \cancel{y_3} + \cancel{y_4} + \cancel{y_5} + \cancel{y_6} + \cancel{y_7} + \cancel{y_8} + \cancel{y_9} + \dots$

Trail: $x_1^d\ \overline{x_2}^d\ x_3^p\ \overline{x_4}^d\ x_5^p\ \overline{y_9}^a\ y_1^p\ y_3^p$

Found disjoint local "cores"

Core 1: $y_3 + y_5 + y_6 \geq 1$

Core 2: $y_1 + y_2 + y_4 + y_7 + y_8 \geq 2$

LOOKAHEAD WITH LITERAL UNLOCKING (BY EXAMPLE)

$\mathcal{O}^t = \cancel{y_1} + \cancel{y_2} + \cancel{y_3} + \cancel{y_4} + y_5 + y_6 + \cancel{y_7} + \cancel{y_8} + \cancel{y_9} + \dots$

Trail: $x_1^d\ \overline{x_2}^d\ x_3^p\ \overline{x_4}^d\ x_5^p\ \overline{y_9}^a\ y_1^p\ y_3^p$

Found disjoint local "cores"

Core 1: $y_3 + \cancel{y_5} + \cancel{y_6} \geq 1$

"$\{y_9\}$ unlocks Core 1 on $\{y_3\}$"

Core 2: $y_1 + y_2 + y_4 + y_7 + y_8 \geq 2$

LOOKAHEAD WITH LITERAL UNLOCKING (BY EXAMPLE)

$\mathcal{O}^t = \not{y_1} + \not{y_2} + \not{y_3} + \not{y_4} + \not{y_5} + \not{y_6} + \not{y_7} + \not{y_8} + \not{y_9} + \ldots$

Trail: $x_1^d\ \overline{x_2}^d\ x_3^p\ \overline{x_4}^d\ x_5^p\ \overline{y_9}^a\ y_1^p\ y_3^p\ \overline{y_5}^a$

Found disjoint local "cores"

Core 1: $y_3 + \not{y_5} + \not{y_6} \geq 1$

"$\{y_9\}$ unlocks Core 1 on $\{y_3\}$"

Core 2: $y_1 + y_2 + y_4 + y_7 + y_8 \geq 2$

LOOKAHEAD WITH LITERAL UNLOCKING (BY EXAMPLE)

$\mathcal{O}^t = \cancel{y_1} + \cancel{y_2} + \cancel{y_3} + \cancel{y_4} + \cancel{y_5} + \cancel{y_6} + \cancel{y_7} + \cancel{y_8} + \cancel{y_9} + \ldots$

Trail: $x_1^d \; \overline{x_2}^d \; x_3^p \; \overline{x_4}^d \; x_5^p \; \overline{y_9}^a \; y_1^p \; y_3^p \; \overline{y_5}^a \; \overline{y_6}^a \; y_7^p$

Found disjoint local "cores"

Core 1: $y_3 + \cancel{y_5} + \cancel{y_6} \geq 1$

"$\{y_9\}$ unlocks Core 1 on $\{y_3\}$"

Core 2: $y_1 + y_2 + y_4 + y_7 + y_8 \geq 2$

LOOKAHEAD WITH LITERAL UNLOCKING (BY EXAMPLE)

$\mathcal{O}^t = \cancel{y_1} + y_2 + \cancel{y_3} + y_4 + \cancel{y_5} + \cancel{y_6} + \cancel{y_7} + y_8 + \cancel{y_9} + \ldots$

Trail: $x_1^d \; \overline{x_2}^d \; x_3^p \; \overline{x_4}^d \; x_5^p \; \overline{y_9}^a \; y_1^p \; y_3^p \; \overline{y_5}^a \; \overline{y_6}^a \; y_7^p$

Found disjoint local "cores"

Core 1: $y_3 + \cancel{y_5} + \cancel{y_6} \geq 1$

"$\{y_9\}$ unlocks Core 1 on $\{y_3\}$"

Core 2: $y_1 + \cancel{y_2} + \cancel{y_4} + y_7 + \cancel{y_8} \geq 2$

"$\{y_9, y_5, y_6\}$ unlocks Core 2 on $\{y_1, y_7\}$"

LOOKAHEAD WITH LITERAL UNLOCKING (BY EXAMPLE)

$\mathcal{O}^t = \cancel{y_1} + \cancel{y_2} + \cancel{y_3} + \cancel{y_4} + \cancel{y_5} + \cancel{y_6} + \cancel{y_7} + y_8 + \cancel{y_9} + \ldots$

Trail: $x_1^d\ \overline{x_2}^d\ x_3^p\ \overline{x_4}^d\ x_5^p\ \overline{y_9}^a\ y_1^p\ y_3^p\ \overline{y_5}^a\ \overline{y_6}^a\ y_7^p\ \overline{y_2}^a\ \bot$

Found disjoint local "cores"

Core 1: $y_3 + \cancel{y_5} + \cancel{y_6} \geq 1$

"$\{y_9\}$ unlocks Core 1 on $\{y_3\}$"

Core 2: $y_1 + \cancel{y_2} + \cancel{y_4} + y_7 + \cancel{y_8} \geq 2$

"$\{y_9, y_5, y_6\}$ unlocks Core 2 on $\{y_1, y_7\}$"

New core: $y_9 + y_5 + y_6 + y_2 \geq 1$

LOOKAHEAD WITH LITERAL UNLOCKING (BY EXAMPLE)

$\mathcal{O}^t = \cancel{y_1} + \cancel{y_2} + \cancel{y_3} + \cancel{y_4} + \cancel{y_5} + \cancel{y_6} + \cancel{y_7} + \cancel{y_8} + \cancel{y_9} + \dots$

Trail: $x_1^d\ \overline{x_2}^d\ x_3^p\ \overline{x_4}^d\ x_5^p\ \overline{y_9}^a\ y_1^p\ y_3^p\ \overline{y_5}^a\ \overline{y_6}^a\ y_7^p\ \overline{y_2}^a\ \perp$

Found disjoint local "cores"

Core 1: $y_3 + \cancel{y_5} + \cancel{y_6} \geq 1$

"$\{y_9\}$ unlocks Core 1 on $\{y_3\}$"

Core 2: $y_1 + \cancel{y_2} + \cancel{y_4} + y_7 + \cancel{y_8} \geq 2$

"$\{y_9, y_5, y_6\}$ unlocks Core 2 on $\{y_1, y_7\}$"

New core: $y_9 + y_5 + y_6 + y_2 \geq 1$

Conclusion $\sum_{i=1}^{9} y_i \geq 4$ **?**

LOOKAHEAD WITH LITERAL UNLOCKING (BY EXAMPLE)

$\mathcal{O}^t = \cancel{y_1} + \cancel{y_2} + \cancel{y_3} + \cancel{y_4} + \cancel{y_5} + \cancel{y_6} + \cancel{y_7} + \cancel{y_8} + \cancel{y_9} + \dots$

Trail: $x_1^d\ \overline{x_2}^d\ x_3^p\ \overline{x_4}^d\ x_5^p\ \overline{y_9}^a\ y_1^p\ y_3^p\ \overline{y_5}^a\ \overline{y_6}^a\ y_7^p\ \overline{y_2}^a\ \perp$

Found disjoint local "cores"

Core 1: $y_3 + y_5 + y_6 \geq 1$

"$\{y_9\}$ unlocks Core 1 on $\{y_3\}$"

Core 2: $y_1 + y_2 + y_4 + y_7 + y_8 \geq 2$

"$\{y_9, y_5, y_6\}$ unlocks Core 2 on $\{y_1, y_7\}$"

New core: $y_9 + y_5 + y_6 + y_2 \geq 1$

Conclusion $\sum_{i=1}^{9} y_i \geq 4$ **?**

LOOKAHEAD WITH LITERAL UNLOCKING (BY EXAMPLE)

$\mathcal{O}^t = \cancel{y_1} + \cancel{y_2} + \cancel{y_3} + \cancel{y_4} + \cancel{y_5} + \cancel{y_6} + \cancel{y_7} + \cancel{y_8} + \cancel{y_9} + \ldots$

Trail: $x_1^d \; \overline{x_2}^d \; x_3^p \; \overline{x_4}^d \; x_5^p \; \overline{y_9}^a \; y_1^p \; y_3^p \; \overline{y_5}^a \; \overline{y_6}^a \; y_7^p \; \overline{y_2}^a \; \perp$

Found disjoint local "cores"

Core 1: $\overline{y_9} \rightarrow y_3 + y_5 + y_6 \geq 1$

"$\{y_9\}$ unlocks Core 1 on $\{y_3\}$"

Core 2: $y_1 + y_2 + y_4 + y_7 + y_8 \geq 2$

"$\{y_9, y_5, y_6\}$ unlocks Core 2 on $\{y_1, y_7\}$"

New core: $\cancel{y_9} + y_5 + y_6 + y_2 \geq 1$

Conclusion $\sum_{i=1}^{9} y_i \geq 4$ **?**

LOOKAHEAD WITH LITERAL UNLOCKING (BY EXAMPLE)

$\mathcal{O}^t = \not{y_1} + \not{y_2} + \not{y_3} + \not{y_4} + \not{y_5} + \not{y_6} + \not{y_7} + \not{y_8} + \not{y_9} + \ldots$

Trail: $x_1^d\ \overline{x_2}^d\ x_3^p\ \overline{x_4}^d\ x_5^p\ \overline{y_9}^a\ y_1^p\ y_3^p\ \overline{y_5}^a\ \overline{y_6}^a\ y_7^p\ \overline{y_2}^a\ \perp$

Found disjoint local “cores”

Core 1: $\overline{y_9} \rightarrow y_3 + \not{y_5} + \not{y_6} \geq 1$

“$\{y_9\}$ unlocks Core 1 on $\{y_3\}$”

Core 2: $\overline{y_9} \wedge \overline{y_5} + \overline{y_6} \rightarrow y_1 + y_2 + y_4 + y_7 + y_8 \geq 2$

“$\{y_9, y_5, y_6\}$ unlocks Core 2 on $\{y_1, y_7\}$”

New core: $\not{y_9} + \not{y_5} + \not{y_6} + y_2 \geq 1$

Conclusion $\sum_{i=1}^{9} y_i \geq 4$ **?**

LOOKAHEAD WITH LITERAL UNLOCKING (BY EXAMPLE)

$\mathcal{O}^t = \cancel{y_1} + \cancel{y_2} + \cancel{y_3} + \cancel{y_4} + \cancel{y_5} + \cancel{y_6} + \cancel{y_7} + \cancel{y_8} + \cancel{y_9} + \ldots$

Trail: $x_1^d\ \overline{x_2}^d\ x_3^p\ \overline{x_4}^d\ x_5^p\ \overline{y_9}^a\ y_1^p\ y_3^p\ \overline{y_5}^a\ \overline{y_6}^a\ y_7^p\ \overline{y_2}^a\ \bot$

Found disjoint local "cores"

Core 1: $\overline{y_9} \rightarrow y_3 + \cancel{y_5} + \cancel{y_6} \geq 1$

"$\{y_9\}$ unlocks Core 1 on $\{y_3\}$"

Core 2: $\overline{y_9} \wedge \overline{y_5} + \overline{y_6} \rightarrow y_1 + y_2 + y_4 + y_7 + y_8 \geq 2$

"$\{y_9, y_5, y_6\}$ unlocks Core 2 on $\{y_1, y_7\}$"

New core: $\cancel{y_9} + \cancel{y_5} + \cancel{y_6} + y_2 \geq 1$

Conclusion $\sum_{i=1}^{9} y_i \geq 4$

PROOF LOGGING LITERAL UNLOCKING

Trail: $x_1^d\ \overline{x_2}^d\ x_3^p\ \overline{x_4}^d\ x_5^p\ \overline{y_9}^a\ y_1^p\ y_3^p\ \overline{y_5}^a\ \overline{y_6}^a\ y_7^p\ \overline{y_2}^a\ \bot$

Found disjoint local "cores":

Core 1: $\overline{y_9} \rightarrow y_3 + \cancel{y_5} + \cancel{y_6} \geq 1$

"$\{y_9\}$ unlocks Core 1 on $\{y_3\}$"

Core 2: $\overline{y_9} \wedge \overline{y_5} \wedge \overline{y_6} \rightarrow y_1 + y_2 + y_4 + y_7 + y_8 \geq 2$

"$\{y_9, y_5, y_6\}$ unlocks Core 2 on $\{y_1, y_7\}$"

New core: $\cancel{y_9} + \cancel{y_5} + \cancel{y_6} + y_2 \geq 1$

To Derive: $\sum_{i=1}^{9} y_i \geq 4$

PROOF LOGGING LITERAL UNLOCKING

Trail: $x_1^d\ \overline{x_2}^d\ x_3^p\ \overline{x_4}^d\ x_5^p\ \overline{y_9}^a\ y_1^p\ y_3^p\ \overline{y_5}^a\ \overline{y_6}^a\ y_7^p\ \overline{y_2}^a\ \bot$

Found disjoint local "cores":

Core 1: $\overline{y_9} \rightarrow y_3 + \cancel{y_5} + \cancel{y_6} \geq 1$

"$\{y_9\}$ unlocks Core 1 on $\{y_3\}$"

$y_9 + y_3 \geq 1$

Core 2: $\overline{y_9} \wedge \overline{y_5} \wedge \overline{y_6} \rightarrow y_1 + y_2 + y_4 + y_7 + y_8 \geq 2$

"$\{y_9, y_5, y_6\}$ unlocks Core 2 on $\{y_1, y_7\}$"

$y_9 + y_5 + y_6 + y_1 \geq 1$

$y_9 + y_5 + y_6 + y_7 \geq 1$

New core: $\cancel{y_9} + \cancel{y_5} + \cancel{y_6} + y_2 \geq 1$

$y_9 + y_5 + y_6 + y_2 \geq 1$

To Derive: $\sum_{i=1}^{9} y_i \geq 4$

PROOF LOGGING LITERAL UNLOCKING

Trail: $x_1^d\ \overline{x_2}^d\ x_3^p\ \overline{x_4}^d\ x_5^p\ \overline{y_9}^a\ y_1^p\ y_3^p\ \overline{y_5}^a\ \overline{y_6}^a\ y_7^p\ \overline{y_2}^a\ \bot$

Found disjoint local "cores":

Core 1: $\overline{y_9} \rightarrow y_3 + \cancel{y_5} + \cancel{y_6} \geq 1$

"$\{y_9\}$ unlocks Core 1 on $\{y_3\}$"

$y_9 + y_3 \geq 1$ (RUP)

Core 2: $\overline{y_9} \wedge \overline{y_5} \wedge \overline{y_6} \rightarrow y_1 + y_2 + y_4 + y_7 + y_8 \geq 2$

"$\{y_9, y_5, y_6\}$ unlocks Core 2 on $\{y_1, y_7\}$"

$y_9 + y_5 + y_6 + y_1 \geq 1$ (RUP)

$y_9 + y_5 + y_6 + y_7 \geq 1$ (RUP)

New core: $\cancel{y_9} + \cancel{y_5} + \cancel{y_6} + y_2 \geq 1$

$y_9 + y_5 + y_6 + y_2 \geq 1$ (RUP)

To Derive: $\sum_{i=1}^{9} y_i \geq 4$

PROOF LOGGING LITERAL UNLOCKING

Trail: $x_1^d\ \overline{x_2}^d\ x_3^p\ \overline{x_4}^d\ x_5^p\ \overline{y_9}^a\ y_1^p\ y_3^p\ \overline{y_5}^a\ \overline{y_6}^a\ y_7^p\ \overline{y_2}^a\ \bot$

Notation:

$L = \{y_9\}$

Found disjoint local "cores":

Core 1: $\overline{y_9} \rightarrow y_3 + \not{y_5} + \not{y_6} \geq 1$

$U_1 = \{y_3\},\ R_1 = \{y_5, y_6\}$

"$\{y_9\}$ unlocks Core 1 on $\{y_3\}$"

$y_9 + y_3 \geq 1$ (RUP)

Core 2: $\overline{y_9} \wedge \overline{y_5} \wedge \overline{y_6} \rightarrow y_1 + y_2 + y_4 + y_7 + y_8 \geq 2$

$U_2 = \{y_1, y_7\},\ R_2 = \{y_2, y_4, y_8\}$

"$\{y_9, y_5, y_6\}$ unlocks Core 2 on $\{y_1, y_7\}$"

$y_9 + y_5 + y_6 + y_1 \geq 1$ (RUP)

$y_9 + y_5 + y_6 + y_7 \geq 1$ (RUP)

New core: $\not{y_9} + \not{y_5} + \not{y_6} + y_2 \geq 1$

$y_9 + y_5 + y_6 + y_2 \geq 1$ (RUP)

To Derive: $\sum_{i=1}^{9} y_i \geq 4$

PROOF LOGGING LITERAL UNLOCKING

Trail: $x_1^d\ \overline{x_2}^d\ x_3^p\ \overline{x_4}^d\ x_5^p\ \overline{y_9}^a\ y_1^p\ y_3^p\ \overline{y_5}^a\ \overline{y_6}^a\ y_7^p\ \overline{y_2}^a\ \bot$

Notation:

Found disjoint local "cores":

$L = \{y_9\}$

Core 1: $\overline{y_9} \rightarrow y_3 + \cancel{y_5} + \cancel{y_6} \geq 1$

$U_1 = \{y_3\},\ R_1 = \{y_5, y_6\}$

"$\{y_9\}$ unlocks Core 1 on $\{y_3\}$"

"L unlocks Core 1 on U_1"

$y_9 + y_3 \geq 1$ (RUP)

Core 2: $\overline{y_9} \wedge \overline{y_5} \wedge \overline{y_6} \rightarrow y_1 + y_2 + y_4 + y_7 + y_8 \geq 2$

$U_2 = \{y_1, y_7\},\ R_2 = \{y_2, y_4, y_8\}$

"$\{y_9, y_5, y_6\}$ unlocks Core 2 on $\{y_1, y_7\}$"

"$L \cup R_1$ unlocks Core 2 on U_2"

$y_9 + y_5 + y_6 + y_1 \geq 1$ (RUP)

$y_9 + y_5 + y_6 + y_7 \geq 1$ (RUP)

New core: $\cancel{y_9} + \cancel{y_5} + \cancel{y_6} + y_2 \geq 1$

$y_9 + y_5 + y_6 + y_2 \geq 1$ (RUP)

To Derive: $\sum_{i=1}^{9} y_i \geq 4$

PROOF LOGGING LITERAL UNLOCKING

Trail: $x_1^d\ \overline{x_2}^d\ x_3^p\ \overline{x_4}^d\ x_5^p\ \overline{y_9}^a\ y_1^p\ y_3^p\ \overline{y_5}^a\ \overline{y_6}^a\ y_7^p\ \overline{y_2}^a\ \bot$

Notation:

Found disjoint local "cores":

$L = \{y_9\}$

Core 1: $\overline{y_9} \rightarrow y_3 + \cancel{y_5} + \cancel{y_6} \geq 1$

$U_1 = \{y_3\},\ R_1 = \{y_5, y_6\}$

"$\{y_9\}$ unlocks Core 1 on $\{y_3\}$"

"L unlocks Core 1 on U_1"

$y_9 + y_3 \geq 1$ (RUP)

$L + y_3 \geq 1$

Core 2: $\overline{y_9} \wedge \overline{y_5} \wedge \overline{y_6} \rightarrow y_1 + y_2 + y_4 + y_7 + y_8 \geq 2$

$U_2 = \{y_1, y_7\},\ R_2 = \{y_2, y_4, y_8\}$

"$\{y_9, y_5, y_6\}$ unlocks Core 2 on $\{y_1, y_7\}$"

"$L \cup R_1$ unlocks Core 2 on U_2"

$y_9 + y_5 + y_6 + y_1 \geq 1$ (RUP)

$L + R_1 + y_1 \geq 1$

$y_9 + y_5 + y_6 + y_7 \geq 1$ (RUP)

$L + R_1 + y_7 \geq 1$

New core: $\cancel{y_9} + \cancel{y_5} + \cancel{y_6} + y_2 \geq 1$

$y_9 + y_5 + y_6 + y_2 \geq 1$ (RUP)

$L + R_1 + R_2 \geq 1$

To Derive: $\sum_{i=1}^{9} y_i \geq 4$

PROOF LOGGING LITERAL UNLOCKING

Trail: $x_1^d\ \overline{x_2}^d\ x_3^p\ \overline{x_4}^d\ x_5^p\ \overline{y_9}^a\ y_1^p\ y_3^p\ \overline{y_5}^a\ \overline{y_6}^a\ y_7^p\ \overline{y_2}^a\ \perp$

Notation:

Found disjoint local "cores":

$L = \{y_9\}$

Core 1: $\overline{y_9} \rightarrow y_3 + \cancel{y_5} + \cancel{y_6} \geq 1$

$U_1 = \{y_3\},\ R_1 = \{y_5, y_6\}$

"$\{y_9\}$ unlocks Core 1 on $\{y_3\}$"

"L unlocks Core 1 on U_1"

$y_9 + y_3 \geq 1$ (RUP)

$L + y_3 \geq 1$

Core 2: $\overline{y_9} \wedge \overline{y_5} \wedge \overline{y_6} \rightarrow y_1 + y_2 + y_4 + y_7 + y_8 \geq 2$

$U_2 = \{y_1, y_7\},\ R_2 = \{y_2, y_4, y_8\}$

"$\{y_9, y_5, y_6\}$ unlocks Core 2 on $\{y_1, y_7\}$"

"$L \cup R_1$ unlocks Core 2 on U_2"

$y_9 + y_5 + y_6 + y_1 \geq 1$ (RUP)

$L + R_1 + y_1 \geq 1$

$y_9 + y_5 + y_6 + y_7 \geq 1$ (RUP)

$L + R_1 + y_7 \geq 1$

New core: $\cancel{y_9} + \cancel{y_5} + \cancel{y_6} + y_2 \geq 1$

$y_9 + y_5 + y_6 + y_2 \geq 1$ (RUP)

$L + R_1 + R_2 \geq 1$

To Derive: $\sum_{i=1}^{9} y_i \geq 4$

$L + (\sum_i U_i + R_i) \geq \sum_i b_i + 1$

PROOF LOGGING LITERAL UNLOCKING

From the constraints

$$L_i \geq b_i \ (\forall 1 \leq i \leq k), \qquad L + \sum_{j<i} R_j + \ell \geq 1 \ (\forall 1 \leq i \leq k, \ell \in U_i), \qquad L + \sum_{j} R_j \geq 1$$

we derive

$$L + \sum_{j<i} R_j + \sum_{j \geq i} L_j \geq 1 + \sum_{j \geq i} b_j$$

for each $i \in \{1, \ldots, k+1\}$.

PROOF LOGGING LITERAL UNLOCKING

To Derive: $L + \sum_{j<i} R_j + \sum_{j\geq i} L_j \geq 1 + \sum_{j\geq i} b_j$.

PROOF LOGGING LITERAL UNLOCKING

To Derive: $L + \sum_{j<i} R_j + \sum_{j \geq i} L_j \geq 1 + \sum_{j \geq i} b_j$. By **induction** on i.

PROOF LOGGING LITERAL UNLOCKING

To Derive: $L + \sum_{j<i} R_j + \sum_{j \geq i} L_j \geq 1 + \sum_{j \geq i} b_j$. By **induction** on i.

For $i = k + 1$ (base case):

$$L + \sum_j R_j \geq 1$$

PROOF LOGGING LITERAL UNLOCKING

To Derive: $L + \sum_{j<i} R_j + \sum_{j\geq i} L_j \geq 1 + \sum_{j\geq i} b_j$. By **induction** on i.

For $i = k + 1$ (base case):

$$L + \sum_{j} R_j \geq 1$$

For i between 1 and $k - 1$: Cutting Planes Derivation from IH:

$$L + \sum_{j<i+1} R_j + \sum_{j\geq i+1} L_j \geq 1 + \sum_{j\geq i+1} b_j$$

PROOF LOGGING LITERAL UNLOCKING

To Derive: $L + \sum_{j<i} R_j + \sum_{j \geq i} L_j \geq 1 + \sum_{j \geq i} b_j$. By **induction** on i.

For $i = k + 1$ (base case):

$$L + \sum_j R_j \geq 1$$

For i between 1 and $k - 1$: Cutting Planes Derivation from IH:

$$L + \sum_{j<i+1} R_j + \sum_{j \geq i+1} L_j \geq 1 + \sum_{j \geq i+1} b_j$$

For $i = 1$ (New cardinality constraint!):

$$L + \sum_j L_j \geq 1 + \sum_j b_j$$

OUTLINE OF THIS PRESENTATION

- MaxSAT and how to proof log it
- An introduction to the VeriPB proof system.
- MaxCDCL: Branch-and-Bound with clause learning
- Unweighted MaxCDCL revisited with literal unlocking
- Solution-Improving Constraint using Binary Decision Diagram (BDD) encoding
- Conclusions & Future work

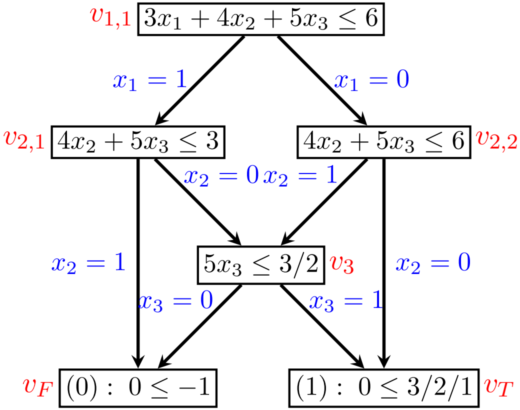
MAXCDCL'S USAGE OF BDDS

MaxCDCL $\cup$ Solution-Improving: MaxCDCL encodes solution-improving constraint

MAXCDCL'S USAGE OF BDDS

MaxCDCL ∪ Solution-Improving: MaxCDCL encodes solution-improving constraint

Binary Decision Diagram:

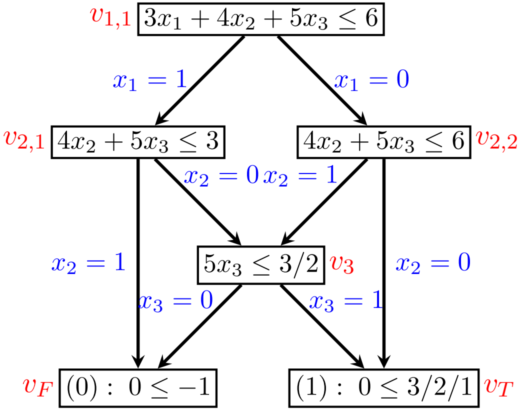


MAXCDCL'S USAGE OF BDDS

MaxCDCL ∪ Solution-Improving: MaxCDCL encodes solution-improving constraint

Binary Decision Diagram:

- Every node corresponds with part of the original PB constraint and,

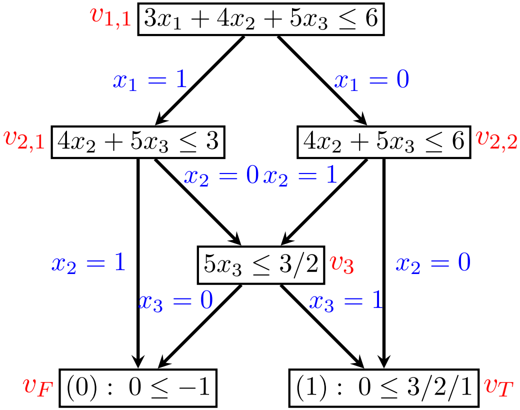


MAXCDCL'S USAGE OF BDDS

MaxCDCL ∪ Solution-Improving: MaxCDCL encodes solution-improving constraint

Binary Decision Diagram:

- Every node corresponds with part of the original PB constraint and,
- Every node propagates based on one decision literal.

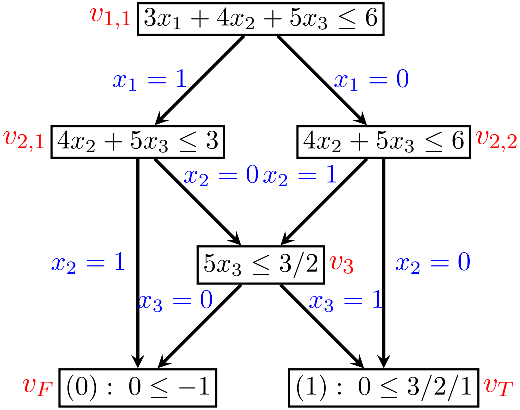


MAXCDCL'S USAGE OF BDDS

MaxCDCL ∪ Solution-Improving: MaxCDCL encodes solution-improving constraint

Binary Decision Diagram:

- Every node corresponds with part of the original PB constraint and,
- Every node propagates based on one decision literal.
- If v_F node is propagated true, then constraint in root is falsified.

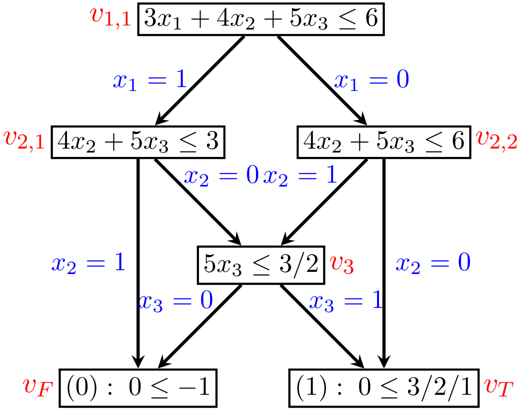


MAXCDCL'S USAGE OF BDDS

MaxCDCL ∪ Solution-Improving: MaxCDCL encodes solution-improving constraint

Introducing fresh variables for each node with meaning:

- E.g., $v_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 6$

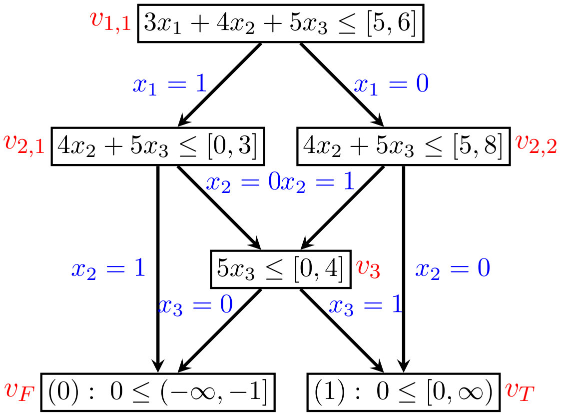


MAXCDCL'S USAGE OF BDDS

MaxCDCL ∪ Solution-Improving: MaxCDCL encodes solution-improving constraint

Introducing fresh variables for each node with meaning:

- E.g., $v_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 6$
- But also $v_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 7$

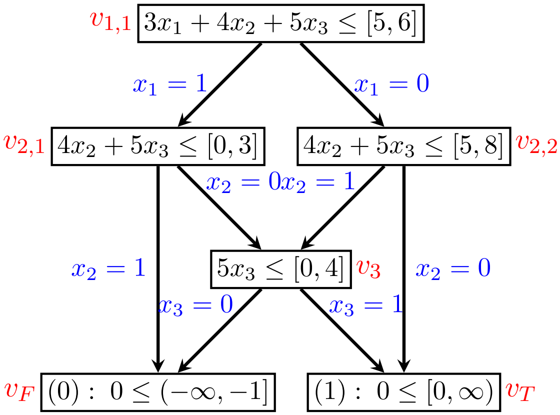


MAXCDCL'S USAGE OF BDDS

MaxCDCL ∪ Solution-Improving: MaxCDCL encodes solution-improving constraint

Introducing fresh variables for each node with meaning:

- E.g., $v_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 6$
- But also $v_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 7$
- Hence, $v_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq [5, 8]$



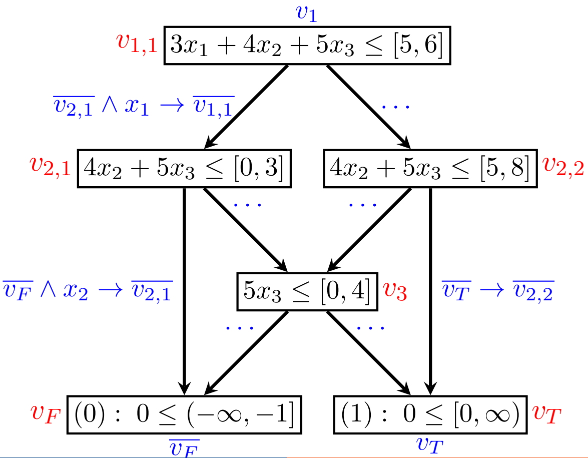
MAXCDCL'S USAGE OF BDDS

MaxCDCL ∪ Solution-Improving: MaxCDCL encodes solution-improving constraint

Introducing fresh variables for each node with meaning:

- E.g., $v_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 6$
- But also $v_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 7$
- Hence, $v_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq [5, 8]$

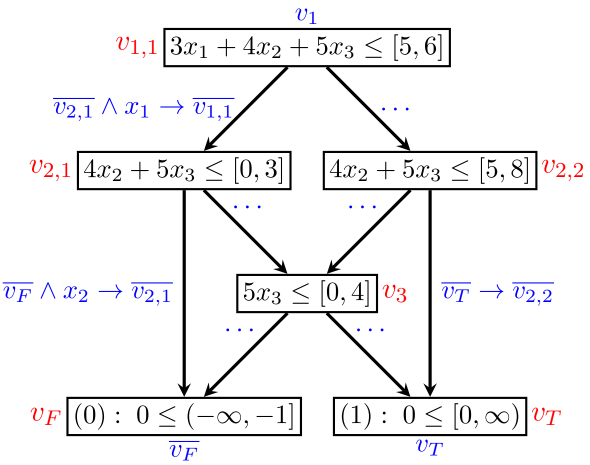
After introducing the variables, clauses are added to the solver.



HOW TO PROOF LOG BDDS?

Step 1: Derive reification of node variables. E.g.,

- $v_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq [5, 8]$
  - $v_{2,2} \rightarrow 4x_2 + 5x_3 \leq 5$
  - $v_{2,2} \leftarrow 4x_2 + 5x_3 \leq 8$



HOW TO PROOF LOG BDDS?

Step 1: Derive reification of node variables. E.g.,

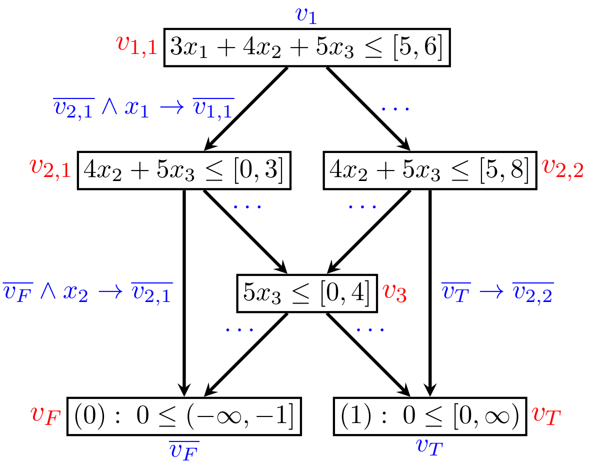
- $v_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq [5, 8]$
  - $v_{2,2} \rightarrow 4x_2 + 5x_3 \leq 5$
  - $v_{2,2} \leftarrow 4x_2 + 5x_3 \leq 8$

by introducing

- $v_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 5$
- $v'_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 8$ (only in proof)

and deriving

- $v'_{2,2} \rightarrow v_{2,2}$



HOW TO PROOF LOG BDDS?

Step 1: Derive reification of node variables. E.g.,

- $v_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq [5, 8]$
  - $v_{2,2} \rightarrow 4x_2 + 5x_3 \leq 5$
  - $v_{2,2} \leftarrow 4x_2 + 5x_3 \leq 8$

by introducing

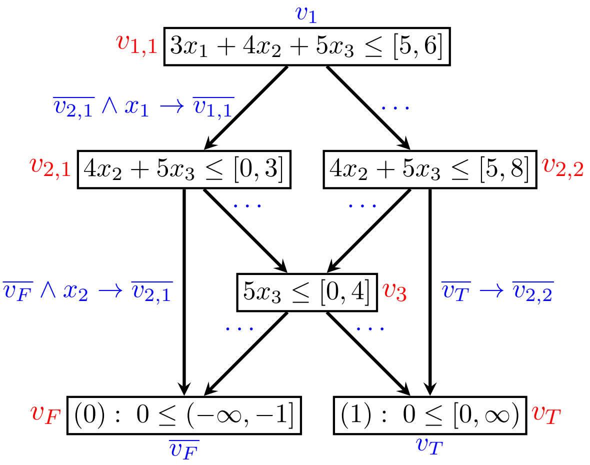
- $v_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 5$
- $v'_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 8$ (only in proof)

and deriving

- $v'_{2,2} \rightarrow v_{2,2}$

Step 2: Derive clauses.

- Straight-forward cutting planes derivation.



INTERMEZZO: PROOF BY CONTRADICTION

Remember definition of Redundance-Based Strengthening:

Definition

A constraint C is redundant with respect to the pseudo-Boolean formula F if there exists a substitution ω, called a witness, such that

$$F \wedge \neg C \models F|_\omega \wedge C|_\omega$$

Proof by contradiction — Take empty witness.

Condition to prove RBS becomes:

$$F \wedge \neg C \models F \wedge C$$

Only one non-trivial proof goal:

$$F \wedge \neg C \wedge \neg C \vdash 0 \geq 1$$

PROVING REIFICATION OF NODE VARIABLES

We have

- $v_{2,2} \rightarrow 4x_2 + 5x_3 \leq 5$
- $v'_{2,2} \leftarrow 4x_2 + 5x_3 \leq 8$

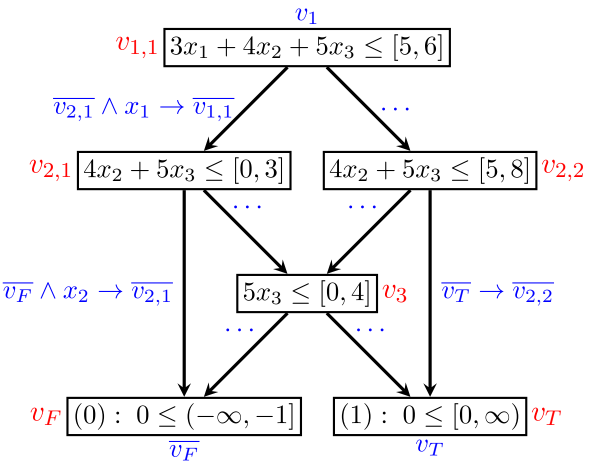
and we want to derive

- $v'_{2,2} \rightarrow v_{2,2}$

If we can prove

- $\overline{x}_2 + \overline{v}'_{2,2} + v_{2,2} \geq 1$
- $x_2 + \overline{v}'_{2,2} + v_{2,2} \geq 1$

then $\overline{v}'_{2,2} + v_{2,2} \geq 1$ follows.



PROVING REIFICATION OF NODE VARIABLES

To derive:

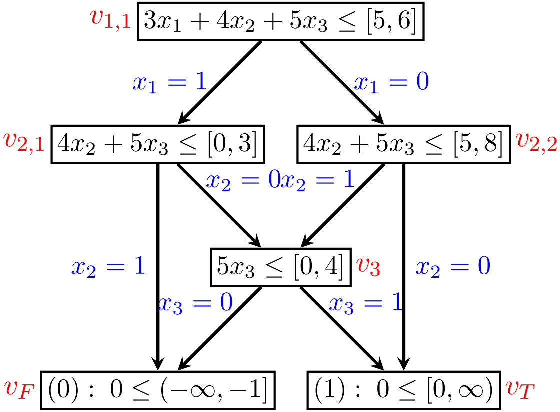
- $\overline{x}_2 + \overline{v}'_{2,2} + v_{2,2} \geq 1$

We have for node $v_{2,2}$:

- $v_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 5$
- $v'_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 8$

For node v_3:

- $v_3 \rightarrow 5x_3 \leq 0$
- $v_3 \leftarrow 5x_3 \leq 4$



PROVING REIFICATION OF NODE VARIABLES (BY CONTRADICTION)

To Derive: $\overline{x}_2 + \overline{v}'_{2,2} + v_{2,2} \geq 1$. We assume the negation, i.e.,

$$x_2 \geq 1, \qquad v'_{2,2} \geq 1, \qquad \overline{v}_{2,2} \geq 1$$

PROVING REIFICATION OF NODE VARIABLES (BY CONTRADICTION)

To Derive: $\overline{x}_2 + \overline{v}'_{2,2} + v_{2,2} \geq 1$. We assume the negation, i.e.,

$$x_2 \geq 1, \qquad v'_{2,2} \geq 1, \qquad \overline{v}_{2,2} \geq 1$$

Constraints already derived:

$$v'_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 8 \qquad v_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 5$$

$$v_3 \leftarrow 5x_3 \leq 4 \qquad v_3 \rightarrow 5x_3 \leq 0$$

PROVING REIFICATION OF NODE VARIABLES (BY CONTRADICTION)

To Derive: $\overline{x}_2 + \overline{v}'_{2,2} + v_{2,2} \geq 1$. We assume the negation, i.e.,

$$x_2 \geq 1, \qquad v'_{2,2} \geq 1, \qquad \overline{v}_{2,2} \geq 1$$

Constraints already derived:

$$v'_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 8 \qquad v_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 5$$

$$v_3 \leftarrow 5x_3 \leq 4 \qquad v_3 \rightarrow 5x_3 \leq 0$$

From $v'_{2,2} \geq 1$: $4x_2 + 5x_3 \leq 8$

Using $x_2 \geq 1$: $5x_3 \leq 4$

Using definition of v_3: $v_3 \geq 1$

PROVING REIFICATION OF NODE VARIABLES (BY CONTRADICTION)

To Derive: $\overline{x}_2 + \overline{v}'_{2,2} + v_{2,2} \geq 1$. We assume the negation, i.e.,

$$x_2 \geq 1, \qquad v'_{2,2} \geq 1, \qquad \overline{v}_{2,2} \geq 1$$

Constraints already derived:

$$v'_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 8 \qquad v_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 5$$

$$v_3 \leftarrow 5x_3 \leq 4 \qquad v_3 \rightarrow 5x_3 \leq 0$$

From $v'_{2,2} \geq 1$: $4x_2 + 5x_3 \leq 8$	From $\overline{v}_{2,2} \geq 1$: $4x_2 + 5x_3 \geq 5 + 1$
Using $x_2 \geq 1$: $5x_3 \leq 4$	Weakening x_2: $5x_3 \geq 2$
Using definition of v_3: $v_3 \geq 1$	Using definition of v_3: $\overline{v}_3 \geq 1$

PROVING REIFICATION OF NODE VARIABLES (BY CONTRADICTION)

To Derive: $\overline{x}_2 + \overline{v}'_{2,2} + v_{2,2} \geq 1$. We assume the negation, i.e.,

$$x_2 \geq 1, \qquad v'_{2,2} \geq 1, \qquad \overline{v}_{2,2} \geq 1$$

Constraints already derived:

$$v'_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 8 \qquad v_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 5$$

$$v_3 \leftarrow 5x_3 \leq 4 \qquad v_3 \rightarrow 5x_3 \leq 0$$

From $v'_{2,2} \geq 1$: $4x_2 + 5x_3 \leq 8$

Using $x_2 \geq 1$: $5x_3 \leq 4$

Using definition of v_3: $v_3 \geq 1$

From $\overline{v}_{2,2} \geq 1$: $4x_2 + 5x_3 \geq 5 + 1$

Weakening x_2: $5x_3 \geq 2$

Using definition of v_3: $\overline{v}_3 \geq 1$

Contradiction.

PROVING REIFICATION OF NODE VARIABLES (BY CONTRADICTION)

To Derive: $\overline{x}_2 + \overline{v}'_{2,2} + v_{2,2} \geq 1$. We assume the negation, i.e.,

$$x_2 \geq 1, \qquad v'_{2,2} \geq 1, \qquad \overline{v}_{2,2} \geq 1$$

Constraints already derived:

$$v'_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 8 \qquad v_{2,2} \leftrightarrow 4x_2 + 5x_3 \leq 5$$

$$v_3 \leftarrow 5x_3 \leq 4 \qquad v_3 \rightarrow 5x_3 \leq 0$$

From $v'_{2,2} \geq 1$: $4x_2 + 5x_3 \leq 8$

Using $x_2 \geq 1$: $5x_3 \leq 4$

Using definition of v_3: $v_3 \geq 1$

From $\overline{v}_{2,2} \geq 1$: $4x_2 + 5x_3 \geq 5 + 1$

Weakening x_2: $5x_3 \geq 2$

Using definition of v_3: $\overline{v}_3 \geq 1$

Contradiction. Same reasoning to obtain $x_2 + \overline{v}'_{2,2} + v_{2,2} \geq 1$.

PROVING REIFICATION OF NODE VARIABLES

We have

- $v_{2,2} \rightarrow 4x_2 + 5x_3 \leq 5$
- $v'_{2,2} \leftarrow 4x_2 + 5x_3 \leq 8$

and we want to derive

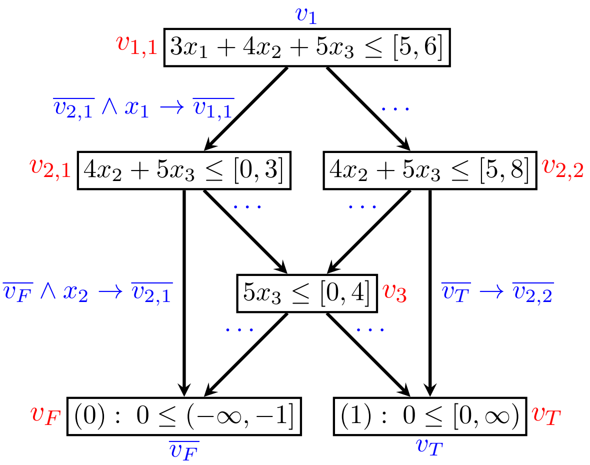
- $v'_{2,2} \rightarrow v_{2,2}$

If we can prove

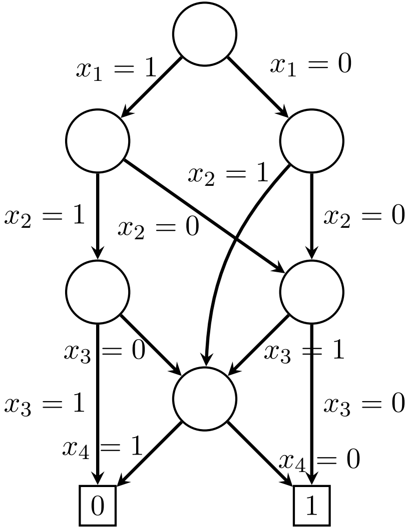
- $\overline{x}_2 + \overline{v}'_{2,2} + v_{2,2} \geq 1$
- $x_2 + \overline{v}'_{2,2} + v_{2,2} \geq 1$

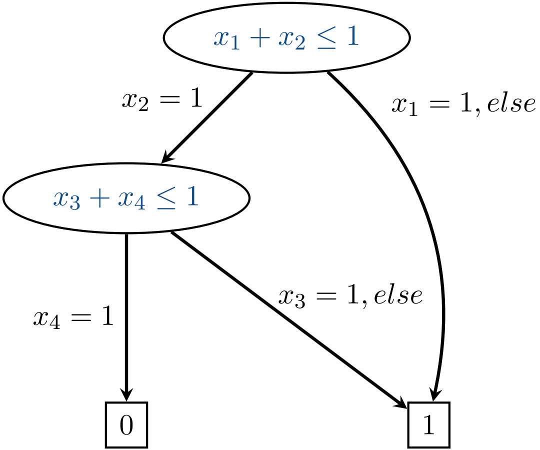
then $\overline{v}'_{2,2} + v_{2,2} \geq 1$ follows.

Clauses: Derived from reification constraints.



MULTI-VALUED DECISION DIAGRAM (MDD)





OUTLINE OF THIS PRESENTATION

- MaxSAT and how to proof log it
- An introduction to the VeriPB proof system.
- MaxCDCL: Branch-and-Bound with clause learning
- Unweighted MaxCDCL revisited with literal unlocking
- Solution-Improving Constraint using Binary Decision Diagram (BDD) encoding
- Conclusions & Future work

WRAPPING UP

Future work:

- Implementation & Experiments
- Implicit Hitting Set solvers

WRAPPING UP

Future work:

- Implementation & Experiments
- Implicit Hitting Set solvers

This talk:

- Proof logging for yet another MaxSAT Solver
  - MaxCDCL: MaxSAT solving by combining Branch-and-Bound and CDCL

WRAPPING UP

Future work:

- Implementation & Experiments
- Implicit Hitting Set solvers

This talk:

- Proof logging for yet another MaxSAT Solver
  - MaxCDCL: MaxSAT solving by combining Branch-and-Bound and CDCL
- Pseudo-Boolean reasoning helps to express MaxSAT algorithms

WRAPPING UP

Future work:

- Implementation & Experiments
- Implicit Hitting Set solvers

This talk:

- Proof logging for yet another MaxSAT Solver
  - MaxCDCL: MaxSAT solving by combining Branch-and-Bound and CDCL
- Pseudo-Boolean reasoning helps to express MaxSAT algorithms
- Proof logging for MaxSAT is possible with VeriPB!!

WRAPPING UP

Future work:

- Implementation & Experiments
- Implicit Hitting Set solvers

This talk:

- Proof logging for yet another MaxSAT Solver
  - MaxCDCL: MaxSAT solving by combining Branch-and-Bound and CDCL
- Pseudo-Boolean reasoning helps to express MaxSAT algorithms
- Proof logging for MaxSAT is possible with VeriPB!!

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

[BBN+24] Jeremias Berg, Bart Bogaerts, Jakob Nordström, Andy Oertel, Tobias Paxian, and Dieter Vandesande. Certifying without loss of generality reasoning in solution-improving maximum satisfiability. In Paul Shaw, editor, *30th International Conference on Principles and Practice of Constraint Programming, CP 2024, September 2-6, 2024, Girona, Spain*, volume 307 of *LIPIcs*, pages 4:1–4:28. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[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.

REFERENCES

[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.

[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.

REFERENCES

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

[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.

[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.

REFERENCES

[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.

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

[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.

[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.

REFERENCES

[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.

[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.

REFERENCES

[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.

[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.

REFERENCES

[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.

[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.

REFERENCES

[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.

[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.

PROOF LOGGING LITERAL UNLOCKING

To Derive: $L + \sum_{j<i} R_j + \sum_{j \geq i} L_j \geq 1 + \sum_{j \geq i} b_j$.

PROOF LOGGING LITERAL UNLOCKING

To Derive: $L + \sum_{j<i} R_j + \sum_{j \geq i} L_j \geq 1 + \sum_{j \geq i} b_j$. By **induction** on i.

PROOF LOGGING LITERAL UNLOCKING

To Derive: $L + \sum_{j<i} R_j + \sum_{j \geq i} L_j \geq 1 + \sum_{j \geq i} b_j$. By **induction** on i.

For $i = k + 1$:

PROOF LOGGING LITERAL UNLOCKING

To Derive: $L + \sum_{j<i} R_j + \sum_{j \geq i} L_j \geq 1 + \sum_{j \geq i} b_j$. By **induction** on i.

For $i = k + 1$:

$$L + \sum_{j} R_j \geq 1$$

PROOF LOGGING LITERAL UNLOCKING

To Derive: $L + \sum_{j<i} R_j + \sum_{j \geq i} L_j \geq 1 + \sum_{j \geq i} b_j$. By **induction** on i.

For i between 1 and $k-1$ (assuming already derived for $i+1$):

PROOF LOGGING LITERAL UNLOCKING

To Derive: $L + \sum_{j<i} R_j + \sum_{j\geq i} L_j \geq 1 + \sum_{j\geq i} b_j$. By **induction** on i.

For i between 1 and $k-1$ (assuming already derived for $i+1$):

Step 0. Induction Hypothesis

$$L + \sum_{j<i+1} R_j + \sum_{j\geq i+1} L_j \geq 1 + \sum_{j\geq i+1} b_j$$

PROOF LOGGING LITERAL UNLOCKING

To Derive: $L + \sum_{j<i} R_j + \sum_{j \geq i} L_j \geq 1 + \sum_{j \geq i} b_j$. By **induction** on i.

For i between 1 and $k-1$ (assuming already derived for $i+1$):

Step 0. Induction Hypothesis

$$L + \sum_{j<i+1} R_j + \sum_{j \geq i+1} L_j \geq 1 + \sum_{j \geq i+1} b_j$$

Step 1. Addition of $L + \sum_{j<i} R_j + \ell \geq 1$ for every $\ell \in U_i$ results in

$$b_i L + b_i \sum_{j<i} R_j + U_i \geq b_i$$

PROOF LOGGING LITERAL UNLOCKING

To Derive: $L + \sum_{j<i} R_j + \sum_{j \geq i} L_j \geq 1 + \sum_{j \geq i} b_j$. By **induction** on i.

For i between 1 and $k-1$ (assuming already derived for $i+1$):

Step 0. Induction Hypothesis

$$L + \sum_{j<i+1} R_j + \sum_{j \geq i+1} L_j \geq 1 + \sum_{j \geq i+1} b_j$$

Step 1. Addition of $L + \sum_{j<i} R_j + \ell \geq 1$ for every $\ell \in U_i$ results in

$$b_i L + b_i \sum_{j<i} R_j + U_i \geq b_i$$

Step 2. Addition with IH gives :

$$(b_i + 1) \cdot L + (b_i + 1) \sum_{j<i} R_j + \sum_{j \geq i} U_j + \sum_{j \geq i} R_j \geq 1 + \sum_{j \geq i} b_j$$

PROOF LOGGING LITERAL UNLOCKING

To Derive: $L + \sum_{j<i} R_j + \sum_{j \geq i} L_j \geq 1 + \sum_{j \geq i} b_j$. By **induction** on i.

For i between 1 and $k-1$ (assuming already derived for $i+1$):

Step 0. Induction Hypothesis

$$L + \sum_{j<i+1} R_j + \sum_{j \geq i+1} L_j \geq 1 + \sum_{j \geq i+1} b_j$$

Step 1. Addition of $L + \sum_{j<i} R_j + \ell \geq 1$ for every $\ell \in U_i$ results in

$$b_i L + b_i \sum_{j<i} R_j + U_i \geq b_i$$

Step 2. Addition with IH gives (with $R_i \cup U_i = L_i$):

$$(b_i + 1) \cdot L + (b_i + 1) \sum_{j<i} R_j + \cancel{\sum_{j \geq i} U_j + \sum_{j \geq i} R_j} + \sum_{j \geq i} L_j \geq 1 + \sum_{j \geq i} b_j$$

PROOF LOGGING LITERAL UNLOCKING

To Derive: $L + \sum_{j<i} R_j + \sum_{j \geq i} L_j \geq 1 + \sum_{j \geq i} b_j$. By **induction** on i.

For i between 1 and $k-1$ (assuming already derived for $i+1$):

Step 0. Induction Hypothesis

$$L + \sum_{j<i+1} R_j + \sum_{j \geq i+1} L_j \geq 1 + \sum_{j \geq i+1} b_j$$

Step 1. Addition of $L + \sum_{j<i} R_j + \ell \geq 1$ for every $\ell \in U_i$ results in

$$b_i L + b_i \sum_{j<i} R_j + U_i \geq b_i$$

Step 2. Addition with IH gives (with $R_i \cup U_i = L_i$):

$$(b_i + 1) \cdot L + (b_i + 1) \sum_{j<i} R_j + \sum_{j \geq i} L_j \geq 1 + \sum_{j \geq i} b_j$$

PROOF LOGGING LITERAL UNLOCKING

To Derive: $L + \sum_{j<i} R_j + \sum_{j\geq i} L_j \geq 1 + \sum_{j\geq i} b_j$. By **induction** on i.

For i between 1 and $k-1$ (assuming already derived for $i+1$):

Step 2. Addition with IH gives (with $R_i \cup U_i = L_i$):

$$(b_i + 1) \cdot L + (b_i + 1) \sum_{j<i} R_j + \sum_{j\geq i} L_j \geq 1 + \sum_{j\geq i} b_j$$

PROOF LOGGING LITERAL UNLOCKING

To Derive: $L + \sum_{j<i} R_j + \sum_{j \geq i} L_j \geq 1 + \sum_{j \geq i} b_j$. By **induction** on i.

For i between 1 and $k-1$ (assuming already derived for $i+1$):

Step 2. Addition with IH gives (with $R_i \cup U_i = L_i$):

$$(b_i + 1) \cdot L + (b_i + 1) \sum_{j<i} R_j + \sum_{j \geq i} L_j \geq 1 + \sum_{j \geq i} b_j$$

Step 3. Multiplying all constraints $L_j \geq b_j$ for $j \geq i$ with b_i gives:

$$b_i \sum_{j \geq i} L_j \geq b_i \sum_{j \geq i} b_j$$

PROOF LOGGING LITERAL UNLOCKING

To Derive: $L + \sum_{j<i} R_j + \sum_{j \geq i} L_j \geq 1 + \sum_{j \geq i} b_j$. By **induction** on i.

For i between 1 and $k-1$ (assuming already derived for $i+1$):

Step 2. Addition with IH gives (with $R_i \cup U_i = L_i$):

$$(b_i + 1) \cdot L + (b_i + 1) \sum_{j<i} R_j + \sum_{j \geq i} L_j \geq 1 + \sum_{j \geq i} b_j$$

Step 3. Multiplying all constraints $L_j \geq b_j$ for $j \geq i$ with b_i gives:

$$b_i \sum_{j \geq i} L_j \geq b_i \sum_{j \geq i} b_j$$

Step 4. Addition of constraints from *Step 2* and *Step 3*:

$$(b_i + 1) \cdot L + (b_i + 1) \sum_{j<i} R_j + (b_i + 1) \sum_{j \geq i} L_j \geq 1 + (b_i + 1) \sum_{j \geq i} b_j$$

PROOF LOGGING LITERAL UNLOCKING

To Derive: $L + \sum_{j<i} R_j + \sum_{j \geq i} L_j \geq 1 + \sum_{j \geq i} b_j$. By **induction** on i.

For i between 1 and $k-1$ (assuming already derived for $i+1$):

Step 4. Addition of constraints from *Step 2* and *Step 3*:

$$(b_i + 1) \cdot L + (b_i + 1) \sum_{j<i} R_j + (b_i + 1) \sum_{j \geq i} L_j \geq 1 + (b_i + 1) \sum_{j \geq i} b_j$$

PROOF LOGGING LITERAL UNLOCKING

To Derive: $L + \sum_{j<i} R_j + \sum_{j \geq i} L_j \geq 1 + \sum_{j \geq i} b_j$. By **induction** on i.

For i between 1 and $k-1$ (assuming already derived for $i+1$):

Step 4. Addition of constraints from *Step 2* and *Step 3*:

$$(b_i + 1) \cdot L + (b_i + 1) \sum_{j<i} R_j + (b_i + 1) \sum_{j \geq i} L_j \geq 1 + (b_i + 1) \sum_{j \geq i} b_j$$

Step 5. Dividing this by $b_{i+1} + 1$ (and rounding the righthand-side up) yields

$$L + \sum_{j<i} R_j + \sum_{j \geq i} L_j \geq 1 + \sum_{j \geq i} b_j$$

INTERMEZZO: PROOF BY CONTRADICTION

Remember definition of Redundance-Based Strengthening:

Definition

A constraint C is redundant with respect to the pseudo-Boolean formula F if and only if there exists a substitution ω, called a witness, such that

$$F \wedge \neg C \models F|_{\omega} \wedge C|_{\omega}$$

INTERMEZZO: PROOF BY CONTRADICTION

Remember definition of Redundance-Based Strengthening:

Definition

A constraint C is redundant with respect to the pseudo-Boolean formula F if and only if there exists a substitution ω, called a witness, such that

$$F \wedge \neg C \models F|_\omega \wedge C|_\omega$$

Proof by contradiction — Take empty witness.

Condition to prove RBS becomes:

$$F \wedge \neg C \models F \wedge C$$

Only one non-trivial proof goal:

$$F \wedge \neg C \wedge \neg C \vdash 0 \geq 1$$

INTERMEZZO: PROOF BY CASE SPLITTING

Suppose we have derived two constraints:

$$a \cdot x + \sum_i b_i l_i \geq B \qquad\qquad a \cdot \overline{x} + \sum_i b_i l_i \geq B$$

And we want to derive the constraint

$$\sum_i b_i l_i \geq B$$

INTERMEZZO: PROOF BY CASE SPLITTING

Suppose we have derived two constraints:

$$a \cdot x + \sum_i b_i l_i \geq B \qquad\qquad a \cdot \overline{x} + \sum_i b_i l_i \geq B$$

And we want to derive the constraint

$$\sum_i b_i l_i \geq B$$

By contradiction. Needed: CP derivation that shows

$$(a \cdot x + \sum_i b_i l_i \geq B) \land (a \cdot \overline{x} + \sum_i b_i l_i \geq B) \land \neg(\sum_i b_i l_i \geq B) \vdash 0 \geq 1$$

INTERMEZZO: PROOF BY CASE SPLITTING

Suppose we have derived two constraints:

$$a \cdot x + \sum_i b_i l_i \geq B \qquad\qquad a \cdot \overline{x} + \sum_i b_i l_i \geq B$$

And we want to derive the constraint

$$\sum_i b_i l_i \geq B$$

By contradiction. Needed: CP derivation that shows

$$(a \cdot x + \sum_i b_i l_i \geq B) \wedge (a \cdot \overline{x} + \sum_i b_i l_i \geq B) \wedge \neg(\sum_i b_i l_i \geq B) \vdash 0 \geq 1$$

After normalization:

$$(a \cdot x + \sum_i b_i l_i \geq B) \wedge (a \cdot \overline{x} + \sum_i b_i l_i \geq B) \wedge (\sum_i b_i l_i \geq \sum_i b_i - B + 1) \vdash 0 \geq 1$$

INTERMEZZO: PROOF BY CASE SPLITTING

To show:

$$(a \cdot x + \sum_i b_i l_i \geq B) \land (a \cdot \overline{x} + \sum_i b_i l_i \geq B) \land (\sum_i b_i \bar{l}_i \geq \sum_i b_i - B + 1) \vdash 0 \geq 1$$

Addition of $(a \cdot x + \sum_i b_i l_i \geq B)$ with $(\sum_i b_i \bar{l}_i \geq \sum_i b_i - B + 1)$ gives

$$a \cdot x + \sum_i b_i l_i + \sum_i b_i \bar{l}_i \geq B + \sum_i b_i - B + 1$$

which is equal to

$$a \cdot x \geq 1$$

After saturation: $x \geq 1$.
Similarly, addition of $(a \cdot \overline{x} + \sum_i b_i l_i \geq B)$ and $(\sum_i b_i \bar{l}_i \geq \sum_i b_i - B + 1)$ and saturation gives

$$\overline{x} \geq 1$$

which is clearly contradiction with $x \geq 1$.