Combinatorial Solving with Provably Correct Results: from SAT to MaxSAT (and Beyond?)

Bart Bogaerts
(thanks to numerous collaborators)
KU Leuven

25/11/2024 – TAASP workshop

OUTLINE

1. Introduction
    1. The Success of Combinatorial Solving (and the Dirty Little Secret...)
    2. Ensuring Correctness with the Help of Proof Logging
2. Proof Logging for SAT
    1. SAT Basics
    2. DPLL and CDCL
    3. Proof System for SAT Proof Logging
3. Pseudo-Boolean Proof Logging
    1. Pseudo-Boolean Constraints and Cutting Planes Reasoning
    2. Pseudo-Boolean Proof Logging for SAT Solving
    3. More Pseudo-Boolean Proof Logging Rules
4. Proof Logging for SAT-Based Optimisation (MaxSAT solving)
    1. Linear SAT-UNSAT Search
    2. Core-Guided Search
    3. Branch-And-Bound Search
5. Conclusion

OUTLINE

COMBINATORIAL SOLVING AND OPTIMISATION

- Revolution last couple of decades in combinatorial solvers for
  - Boolean satisfiability (SAT) solving [BHvMW21][1]
  - Constraint programming (CP) [RvBW06]
  - Mixed integer linear programming (MIP) [AW13, BR07]
- Solve NP-complete problems (or worse) very successfully in practice!
- Except solvers are sometimes wrong... (Even best commercial ones) [BLB10, CKSW13, AGJ+18, GSD19, GS19, BMN22, BBN+23]
- Even get feasibility of solutions wrong (though this should be straightforward!)
- And how to check the absence of solutions?
- Or that a solution is optimal? (Even off-by-one mistakes can snowball into large errors if solver used as subroutine)

[1] See end of slides for all references with bibliographic details

WHAT CAN BE DONE ABOUT SOLVER BUGS?

- **Software testing**
  Hard to get good test coverage for sophisticated solvers
  Inherently can only detect presence of bugs, not absence

WHAT CAN BE DONE ABOUT SOLVER BUGS?

- **Software testing**
  Hard to get good test coverage for sophisticated solvers
  Inherently can only detect presence of bugs, not absence

- **Formal verification**
  Prove that solver implementation adheres to formal specification
  Current techniques cannot scale to this level of complexity

WHAT CAN BE DONE ABOUT SOLVER BUGS?

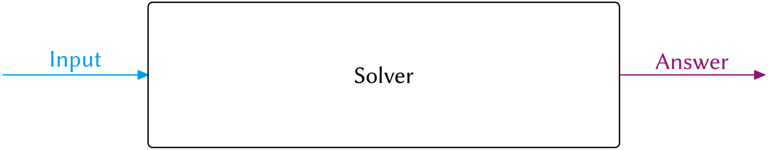
- **Software testing**
  Hard to get good test coverage for sophisticated solvers
  Inherently can only detect presence of bugs, not absence

- **Formal verification**
  Prove that solver implementation adheres to formal specification
  Current techniques cannot scale to this level of complexity

- **Proof logging**
  Make solver certifying [ABM+11, MMNS11] by outputting
  1. not only answer but also
  2. simple, machine-verifiable proof that answer is correct

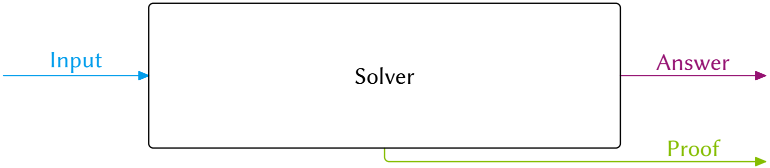
OUTLINE

PROOF LOGGING WITH CERTIFYING SOLVERS: WORKFLOW



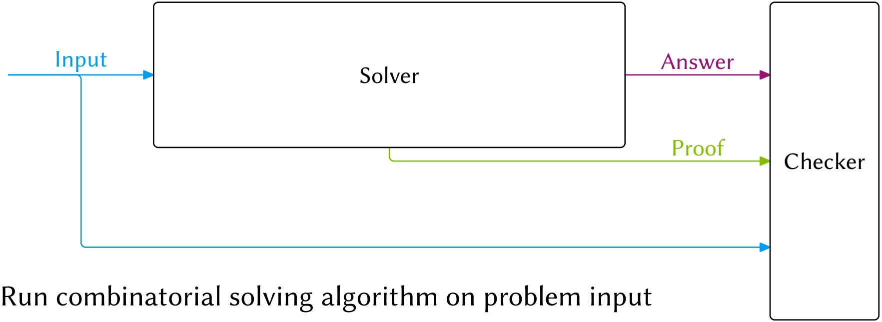
1. Run combinatorial solving algorithm on problem input

PROOF LOGGING WITH CERTIFYING SOLVERS: WORKFLOW



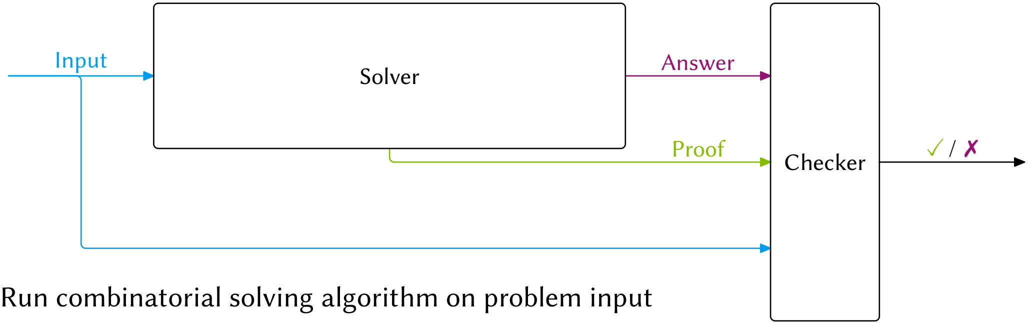
1. Run combinatorial solving algorithm on problem input
2. Get as output not only answer but also proof

PROOF LOGGING WITH CERTIFYING SOLVERS: WORKFLOW



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

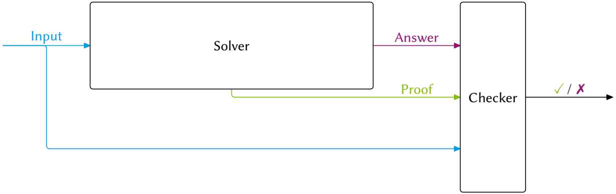
PROOF LOGGING WITH CERTIFYING SOLVERS: WORKFLOW



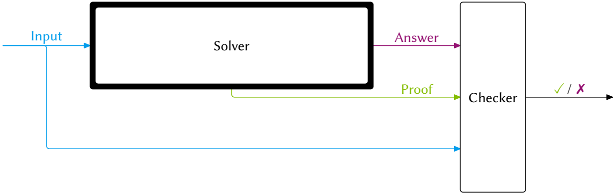
1. Run combinatorial solving algorithm on problem input
2. Get as output not only answer but also proof
3. Feed input + answer + proof to proof checker
4. Verify that proof checker says answer is correct

PROOF LOGGING DESIDERATA

Proof format for certifying solver should be



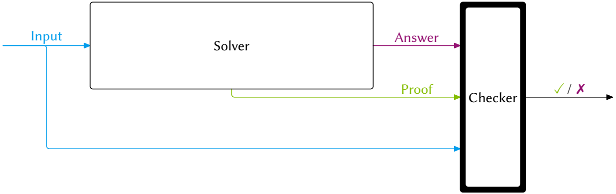
PROOF LOGGING DESIDERATA



Proof format for certifying solver should be

- **very powerful:** minimal overhead for sophisticated reasoning

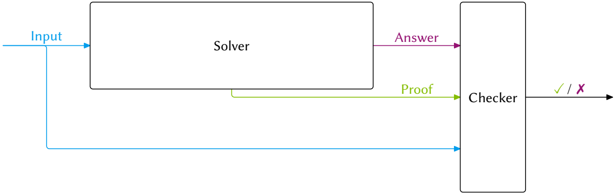
PROOF LOGGING DESIDERATA



Proof format for certifying solver should be

- **very powerful:** minimal overhead for sophisticated reasoning
- **dead simple:** checking correctness of proofs should be trivial

PROOF LOGGING DESIDERATA

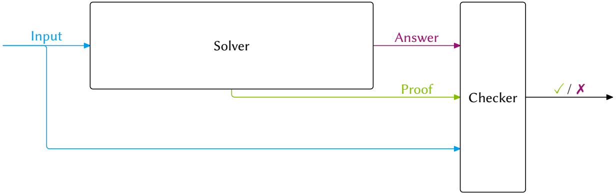


Proof format for certifying solver should be

- **very powerful:** minimal overhead for sophisticated reasoning
- **dead simple:** checking correctness of proofs should be trivial

Clear conflict expressivity vs. simplicity!

PROOF LOGGING DESIDERATA



Proof format for certifying solver should be

- **very powerful:** minimal overhead for sophisticated reasoning
- **dead simple:** checking correctness of proofs should be trivial

Clear conflict expressivity vs. simplicity!

Asking for both perhaps a little bit too good to be true?

TAKE-AWAY MESSAGE

Proof logging for combinatorial optimisation is possible with single, unified method!

TAKE-AWAY MESSAGE

Proof logging for combinatorial optimisation is possible with single, unified method!

- Build on successes in proof logging for SAT solvers with proof formats such as DRAT [HHW13a, HHW13b, WHH14], GRIT [CMS17], LRAT [CHH⁺17], …
- But represent constraints as 0–1 integer linear inequalities
- Formalize reasoning using cutting planes [CCT87] proof system
- Add well-chosen strengthening rules [Goc22, GN21, BGMN23]
- Implemented in VeriPB (`https://gitlab.com/MIAOresearch/software/VeriPB`)

THE SALES PITCH FOR PROOF LOGGING

1. Certifies correctness of computed results
2. Detects errors even if due to compiler bugs, hardware failures, or cosmic rays
3. Provides debugging support during development [EG21, GMM⁺20, KM21, BBN⁺23]
4. Facilitates performance analysis
5. Helps identify potential for further improvements
6. Enables auditability
7. Serves as stepping stone towards explainability

APPLICATIONS OF VERIPB

VERIPB has been used to do proof logging for

- SAT solving (including advanced techniques)
- SAT-based optimisation (MaxSAT) (this talk!)
- Subgraph algorithms
- Constraint programming
- Symmetry and dominance reasoning

in a unified way

OUTLINE

OUTLINE

1. Introduction
    1. The Success of Combinatorial Solving (and the Dirty Little Secret...)
    2. Ensuring Correctness with the Help of Proof Logging
2. Proof Logging for SAT
    1. SAT Basics
    2. DPLL and CDCL
    3. Proof System for SAT Proof Logging
3. Pseudo-Boolean Proof Logging
    1. Pseudo-Boolean Constraints and Cutting Planes Reasoning
    2. Pseudo-Boolean Proof Logging for SAT Solving
    3. More Pseudo-Boolean Proof Logging Rules
4. Proof Logging for SAT-Based Optimisation (MaxSAT solving)
    1. Linear SAT-UNSAT Search
    2. Core-Guided Search
    3. Branch-And-Bound Search
5. Conclusion

THE SAT PROBLEM

- Variable x: takes value **true** ($=1$) or **false** ($=0$)
- Literal ℓ: variable x or its negation $\overline{x}$
- Clause $C = \ell_1 \vee \cdots \vee \ell_k$: disjunction of literals
  (Consider as sets, so no repetitions and order irrelevant)
- Conjunctive normal form (CNF) formula $F = C_1 \wedge \cdots \wedge C_m$: conjunction of clauses

The SAT Problem

Given a CNF formula F, is it satisfiable?

For instance, what about:

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge$$
$$(x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

PROOFS FOR SAT

For satisfiable instances: just specify satisfying assignment

For unsatisfiability: a sequence of clauses (CNF constraints)

- Each clause follows "obviously" from everything we know so far
- Final clause is empty, meaning contradiction (written $\perp$)
- Means original formula must be inconsistent

WHAT IS OBVIOUS? UNIT PROPAGATION

Unit Propagation

Clause C unit propagates ℓ under partial assignment ρ if ρ falsifies all literals in C except ℓ

WHAT IS OBVIOUS? UNIT PROPAGATION

Unit Propagation

Clause C unit propagates ℓ under partial assignment ρ if ρ falsifies all literals in C except ℓ

Example: Unit propagate for $\rho = \{p \mapsto 0, q \mapsto 0\}$ on

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

WHAT IS OBVIOUS? UNIT PROPAGATION

Unit Propagation

Clause C unit propagates ℓ under partial assignment ρ if ρ falsifies all literals in C except ℓ

Example: Unit propagate for $\rho = \{p \mapsto 0, q \mapsto 0\}$ on

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

WHAT IS OBVIOUS? UNIT PROPAGATION

Unit Propagation

Clause C unit propagates ℓ under partial assignment ρ if ρ falsifies all literals in C except ℓ

Example: Unit propagate for $\rho = \{p \mapsto 0, q \mapsto 0\}$ on

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

- $p \vee \overline{u}$ propagates $u \mapsto 0$

WHAT IS OBVIOUS? UNIT PROPAGATION

Unit Propagation

Clause C unit propagates ℓ under partial assignment ρ if ρ falsifies all literals in C except ℓ

Example: Unit propagate for $\rho = \{p \mapsto 0, q \mapsto 0\}$ on

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

- $p \vee \overline{u}$ propagates $u \mapsto 0$
- $q \vee r$ propagates $r \mapsto 1$

WHAT IS OBVIOUS? UNIT PROPAGATION

Unit Propagation

Clause C unit propagates ℓ under partial assignment ρ if ρ falsifies all literals in C except ℓ

Example: Unit propagate for $\rho = \{p \mapsto 0, q \mapsto 0\}$ on

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

- $p \vee \overline{u}$ propagates $u \mapsto 0$
- $q \vee r$ propagates $r \mapsto 1$
- Then $\overline{r} \vee w$ propagates $w \mapsto 1$

WHAT IS OBVIOUS? UNIT PROPAGATION

Unit Propagation

Clause C unit propagates ℓ under partial assignment ρ if ρ falsifies all literals in C except ℓ

Example: Unit propagate for $\rho = \{p \mapsto 0, q \mapsto 0\}$ on

$$(p \lor \overline{u}) \land (q \lor r) \land (\overline{r} \lor w) \land (u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{p} \lor \overline{u})$$

- $p \lor \overline{u}$ propagates $u \mapsto 0$
- $q \lor r$ propagates $r \mapsto 1$
- Then $\overline{r} \lor w$ propagates $w \mapsto 1$
- No further unit propagations

WHAT IS OBVIOUS? UNIT PROPAGATION

Unit Propagation

Clause C unit propagates ℓ under partial assignment ρ if ρ falsifies all literals in C except ℓ

Example: Unit propagate for $\rho = \{p \mapsto 0, q \mapsto 0\}$ on

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

- $p \vee \overline{u}$ propagates $u \mapsto 0$
- $q \vee r$ propagates $r \mapsto 1$
- Then $\overline{r} \vee w$ propagates $w \mapsto 1$
- No further unit propagations

Proof checker should know how to unit propagate until saturation

OUTLINE

1. Introduction
   1. The Success of Combinatorial Solving (and the Dirty Little Secret...)
   2. Ensuring Correctness with the Help of Proof Logging
2. Proof Logging for SAT
   1. SAT Basics
   2. DPLL and CDCL
   3. Proof System for SAT Proof Logging
3. Pseudo-Boolean Proof Logging
   1. Pseudo-Boolean Constraints and Cutting Planes Reasoning
   2. Pseudo-Boolean Proof Logging for SAT Solving
   3. More Pseudo-Boolean Proof Logging Rules
4. Proof Logging for SAT-Based Optimisation (MaxSAT solving)
   1. Linear SAT-UNSAT Search
   2. Core-Guided Search
   3. Branch-And-Bound Search
5. Conclusion

DAVIS-PUTMAN-LOGEMANN-LOVELAND (DPLL)

DPLL [DP60, DLL62]: Assign variables and propagate; backtrack when clause violated

"Proof trace": when backtracking, write negation of guesses made

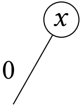
$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

DAVIS-PUTMAN-LOGEMANN-LOVELAND (DPLL)

DPLL [DP60, DLL62]: Assign variables and propagate; backtrack when clause violated

"Proof trace": when backtracking, write negation of guesses made

$$(p \lor \overline{u}) \land (q \lor r) \land (\overline{r} \lor w) \land (u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{p} \lor \overline{u})$$

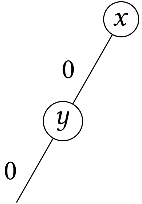


DAVIS-PUTMAN-LOGEMANN-LOVELAND (DPLL)

DPLL [DP60, DLL62]: Assign variables and propagate; backtrack when clause violated

"Proof trace": when backtracking, write negation of guesses made

$$(p \lor \overline{u}) \land (q \lor r) \land (\overline{r} \lor w) \land (u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{p} \lor \overline{u})$$

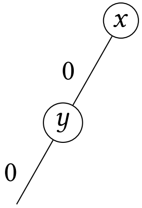


DAVIS-PUTMAN-LOGEMANN-LOVELAND (DPLL)

DPLL [DP60, DLL62]: Assign variables and propagate; backtrack when clause violated

“Proof trace”: when backtracking, write negation of guesses made

$$(p \lor \overline{u}) \land (q \lor r) \land (\overline{r} \lor w) \land (u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{p} \lor \overline{u})$$



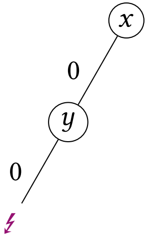
DAVIS-PUTMAN-LOGEMANN-LOVELAND (DPLL)

DPLL [DP60, DLL62]: Assign variables and propagate; backtrack when clause violated

"Proof trace": when backtracking, write negation of guesses made

$$(p \lor \overline{u}) \land (q \lor r) \land (\overline{r} \lor w) \land (u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{p} \lor \overline{u})$$

1. $x \lor y$



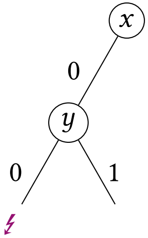
DAVIS-PUTMAN-LOGEMANN-LOVELAND (DPLL)

DPLL [DP60, DLL62]: Assign variables and propagate; backtrack when clause violated

“Proof trace”: when backtracking, write negation of guesses made

$$(p \lor \overline{u}) \land (q \lor r) \land (\overline{r} \lor w) \land (u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{p} \lor \overline{u})$$

1. $x \lor y$



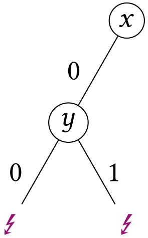
DAVIS-PUTMAN-LOGEMANN-LOVELAND (DPLL)

DPLL [DP60, DLL62]: Assign variables and propagate; backtrack when clause violated

“Proof trace”: when backtracking, write negation of guesses made

$$(p \lor \overline{u}) \land (q \lor r) \land (\overline{r} \lor w) \land (u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{p} \lor \overline{u})$$

1. $x \lor y$
2. $x \lor \overline{y}$



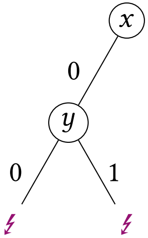
DAVIS-PUTMAN-LOGEMANN-LOVELAND (DPLL)

DPLL [DP60, DLL62]: Assign variables and propagate; backtrack when clause violated

"Proof trace": when backtracking, write negation of guesses made

$$(p \lor \overline{u}) \land (q \lor r) \land (\overline{r} \lor w) \land (u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{p} \lor \overline{u})$$

1. $x \lor y$
2. $x \lor \overline{y}$
3. x



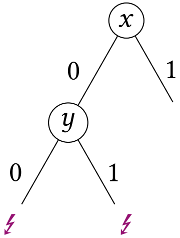
DAVIS-PUTMAN-LOGEMANN-LOVELAND (DPLL)

DPLL [DP60, DLL62]: Assign variables and propagate; backtrack when clause violated

"Proof trace": when backtracking, write negation of guesses made

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

1. $x \vee y$
2. $x \vee \overline{y}$
3. x



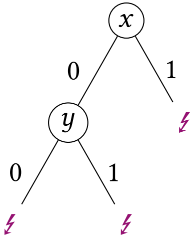
DAVIS-PUTMAN-LOGEMANN-LOVELAND (DPLL)

DPLL [DP60, DLL62]: Assign variables and propagate; backtrack when clause violated

"Proof trace": when backtracking, write negation of guesses made

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

1. $x \vee y$
2. $x \vee \overline{y}$
3. x
4. $\overline{x}$



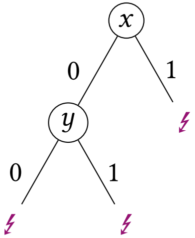
DAVIS-PUTMAN-LOGEMANN-LOVELAND (DPLL)

DPLL [DP60, DLL62]: Assign variables and propagate; backtrack when clause violated

"Proof trace": when backtracking, write negation of guesses made

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

1. $x \vee y$
2. $x \vee \overline{y}$
3. x
4. $\overline{x}$
5. $\perp$



REVERSE UNIT PROPAGATION (RUP)

To make this a proof, need backtrack clauses to be easily verifiable

REVERSE UNIT PROPAGATION (RUP)

To make this a proof, need backtrack clauses to be easily verifiable

Reverse unit propagation (RUP) clause [GN03, Van08]

C is a reverse unit propagation (RUP) clause with respect to F if

- assigning C to false
- then unit propagating on F until saturation
- leads to contradiction

If so, F clearly implies C, and this condition is easy to verify efficiently

REVERSE UNIT PROPAGATION (RUP)

To make this a proof, need backtrack clauses to be easily verifiable

Reverse unit propagation (RUP) clause [GN03, Van08]

C is a reverse unit propagation (RUP) clause with respect to F if

- assigning C to false
- then unit propagating on F until saturation
- leads to contradiction

If so, F clearly implies C, and this condition is easy to verify efficiently

Fact

Backtrack clauses from DPLL solver generate a RUP proof

WHAT ABOUT CONFLICT-DRIVEN CLAUSE LEARNING (CDCL)?

Run CDCL [BS97, MS99, MMZ+01] on our favourite CNF formula:

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

WHAT ABOUT CONFLICT-DRIVEN CLAUSE LEARNING (CDCL)?

Run CDCL [BS97, MS99, MMZ+01] on our favourite CNF formula:

$$(p \lor \overline{u}) \land (q \lor r) \land (\overline{r} \lor w) \land (u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{p} \lor \overline{u})$$

Decision

Free choice to assign value to variable

Notation $p \stackrel{\mathrm{d}}{=} 0$

WHAT ABOUT CONFLICT-DRIVEN CLAUSE LEARNING (CDCL)?

Run CDCL [BS97, MS99, MMZ+01] on our favourite CNF formula:

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

$\boxed{p \overset{\mathrm{d}}{=} 0}$

Decision

Free choice to assign value to variable

Notation $p \overset{\mathrm{d}}{=} 0$

WHAT ABOUT CONFLICT-DRIVEN CLAUSE LEARNING (CDCL)?

Run CDCL [BS97, MS99, MMZ+01] on our favourite CNF formula:

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

$\boxed{p \stackrel{\mathrm{d}}{=} 0}$

Decision

Free choice to assign value to variable

Notation $p \stackrel{\mathrm{d}}{=} 0$

Unit propagation

Forced choice to avoid falsifying clause

Given $p = 0$, clause $p \vee \overline{u}$ forces $u = 0$

Notation $u \stackrel{p \vee \overline{u}}{=} 0$ ($p \vee \overline{u}$ is reason clause)

WHAT ABOUT CONFLICT-DRIVEN CLAUSE LEARNING (CDCL)?

Run CDCL [BS97, MS99, MMZ+01] on our favourite CNF formula:

$$(p \lor \overline{u}) \land (q \lor r) \land (\overline{r} \lor w) \land (u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{p} \lor \overline{u})$$

$\boxed{p \stackrel{\mathrm{d}}{=} 0}$

$u \stackrel{p \lor \overline{u}}{=} 0$

Decision

Free choice to assign value to variable

Notation $p \stackrel{\mathrm{d}}{=} 0$

Unit propagation

Forced choice to avoid falsifying clause

Given $p = 0$, clause $p \lor \overline{u}$ forces $u = 0$

Notation $u \stackrel{p \lor \overline{u}}{=} 0$ ($p \lor \overline{u}$ is reason clause)

WHAT ABOUT CONFLICT-DRIVEN CLAUSE LEARNING (CDCL)?

Run CDCL [BS97, MS99, MMZ[+]01] on our favourite CNF formula:

$$(p \lor \overline{u}) \land (q \lor r) \land (\overline{r} \lor w) \land (u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{p} \lor \overline{u})$$

$\boxed{p \stackrel{\mathrm{d}}{=} 0}$

$u \stackrel{p \lor \overline{u}}{=} 0$

$\boxed{q \stackrel{\mathrm{d}}{=} 0}$

Decision

Free choice to assign value to variable

Notation $p \stackrel{\mathrm{d}}{=} 0$

Unit propagation

Forced choice to avoid falsifying clause

Given $p = 0$, clause $p \lor \overline{u}$ forces $u = 0$

Notation $u \stackrel{p \lor \overline{u}}{=} 0$ ($p \lor \overline{u}$ is reason clause)

Always propagate if possible, otherwise decide

Add to assignment trail

Continue until satisfying assignment or conflict

WHAT ABOUT CONFLICT-DRIVEN CLAUSE LEARNING (CDCL)?

Run CDCL [BS97, MS99, MMZ+01] on our favourite CNF formula:

$$(p \lor \overline{u}) \land (q \lor r) \land (\overline{r} \lor w) \land (u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{p} \lor \overline{u})$$

$\boxed{p \stackrel{\mathrm{d}}{=} 0}$

$u \stackrel{p \lor \overline{u}}{=} 0$

$\boxed{q \stackrel{\mathrm{d}}{=} 0}$

$r \stackrel{q \lor r}{=} 1$

Decision

Free choice to assign value to variable

Notation $p \stackrel{\mathrm{d}}{=} 0$

Unit propagation

Forced choice to avoid falsifying clause

Given $p = 0$, clause $p \lor \overline{u}$ forces $u = 0$

Notation $u \stackrel{p \lor \overline{u}}{=} 0$ ($p \lor \overline{u}$ is reason clause)

Always propagate if possible, otherwise decide

Add to assignment trail

Continue until satisfying assignment or conflict

WHAT ABOUT CONFLICT-DRIVEN CLAUSE LEARNING (CDCL)?

Run CDCL [BS97, MS99, MMZ+01] on our favourite CNF formula:

$$(p \lor \overline{u}) \land (q \lor r) \land (\overline{r} \lor w) \land (u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{p} \lor \overline{u})$$

$\boxed{p \overset{\mathrm{d}}{=} 0}$

$u \overset{p \lor \overline{u}}{=} 0$

$\boxed{q \overset{\mathrm{d}}{=} 0}$

$r \overset{q \lor r}{=} 1$

$w \overset{\overline{r} \lor w}{=} 1$

Decision

Free choice to assign value to variable

Notation $p \overset{\mathrm{d}}{=} 0$

Unit propagation

Forced choice to avoid falsifying clause

Given $p = 0$, clause $p \lor \overline{u}$ forces $u = 0$

Notation $u \overset{p \lor \overline{u}}{=} 0$ ($p \lor \overline{u}$ is reason clause)

Always propagate if possible, otherwise decide

Add to assignment trail

Continue until satisfying assignment or conflict

WHAT ABOUT CONFLICT-DRIVEN CLAUSE LEARNING (CDCL)?

Run CDCL [BS97, MS99, MMZ+01] on our favourite CNF formula:

$$(p \lor \overline{u}) \land (q \lor r) \land (\overline{r} \lor w) \land (u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{p} \lor \overline{u})$$

$\boxed{p \stackrel{\mathrm{d}}{=} 0}$

$u \stackrel{p \lor \overline{u}}{=} 0$

$\boxed{q \stackrel{\mathrm{d}}{=} 0}$

$r \stackrel{q \lor r}{=} 1$

$w \stackrel{\overline{r} \lor w}{=} 1$

$\boxed{x \stackrel{\mathrm{d}}{=} 0}$

Decision

Free choice to assign value to variable

Notation $p \stackrel{\mathrm{d}}{=} 0$

Unit propagation

Forced choice to avoid falsifying clause

Given $p = 0$, clause $p \lor \overline{u}$ forces $u = 0$

Notation $u \stackrel{p \lor \overline{u}}{=} 0$ ($p \lor \overline{u}$ is reason clause)

Always propagate if possible, otherwise decide

Add to assignment trail

Continue until satisfying assignment or conflict

WHAT ABOUT CONFLICT-DRIVEN CLAUSE LEARNING (CDCL)?

Run CDCL [BS97, MS99, MMZ+01] on our favourite CNF formula:

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

$\boxed{p \stackrel{\mathrm{d}}{=} 0}$

$u \stackrel{p \vee \overline{u}}{=} 0$

$\boxed{q \stackrel{\mathrm{d}}{=} 0}$

$r \stackrel{q \vee r}{=} 1$

$w \stackrel{\overline{r} \vee w}{=} 1$

$\boxed{x \stackrel{\mathrm{d}}{=} 0}$

$y \stackrel{u \vee x \vee y}{=} 1$

Decision

Free choice to assign value to variable

Notation $p \stackrel{\mathrm{d}}{=} 0$

Unit propagation

Forced choice to avoid falsifying clause

Given $p = 0$, clause $p \vee \overline{u}$ forces $u = 0$

Notation $u \stackrel{p \vee \overline{u}}{=} 0$ ($p \vee \overline{u}$ is reason clause)

Always propagate if possible, otherwise decide

Add to assignment trail

Continue until satisfying assignment or conflict

WHAT ABOUT CONFLICT-DRIVEN CLAUSE LEARNING (CDCL)?

Run CDCL [BS97, MS99, MMZ+01] on our favourite CNF formula:

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

$\boxed{p \stackrel{\mathrm{d}}{=} 0}$

$u \stackrel{p \vee \overline{u}}{=} 0$

$\boxed{q \stackrel{\mathrm{d}}{=} 0}$

$r \stackrel{q \vee r}{=} 1$

$w \stackrel{\overline{r} \vee w}{=} 1$

$\boxed{x \stackrel{\mathrm{d}}{=} 0}$

$y \stackrel{u \vee x \vee y}{=} 1$

$z \stackrel{x \vee \overline{y} \vee z}{=} 1$

Decision

Free choice to assign value to variable

Notation $p \stackrel{\mathrm{d}}{=} 0$

Unit propagation

Forced choice to avoid falsifying clause

Given $p = 0$, clause $p \vee \overline{u}$ forces $u = 0$

Notation $u \stackrel{p \vee \overline{u}}{=} 0$ ($p \vee \overline{u}$ is reason clause)

Always propagate if possible, otherwise decide

Add to assignment trail

Continue until satisfying assignment or conflict

WHAT ABOUT CONFLICT-DRIVEN CLAUSE LEARNING (CDCL)?

Run CDCL [BS97, MS99, MMZ+01] on our favourite CNF formula:

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

$\boxed{p \stackrel{\mathrm{d}}{=} 0}$

$u \stackrel{p \vee \overline{u}}{=} 0$

$\boxed{q \stackrel{\mathrm{d}}{=} 0}$

$r \stackrel{q \vee r}{=} 1$

$w \stackrel{\overline{r} \vee w}{=} 1$

$\boxed{x \stackrel{\mathrm{d}}{=} 0}$

$y \stackrel{u \vee x \vee y}{=} 1$

$z \stackrel{x \vee \overline{y} \vee z}{=} 1$

$\stackrel{\overline{y} \vee \overline{z}}{\perp}$

Decision

Free choice to assign value to variable

Notation $p \stackrel{\mathrm{d}}{=} 0$

Unit propagation

Forced choice to avoid falsifying clause

Given $p = 0$, clause $p \vee \overline{u}$ forces $u = 0$

Notation $u \stackrel{p \vee \overline{u}}{=} 0$ ($p \vee \overline{u}$ is reason clause)

Always propagate if possible, otherwise decide

Add to assignment trail

Continue until satisfying assignment or conflict

WHAT ABOUT CONFLICT-DRIVEN CLAUSE LEARNING (CDCL)?

Run CDCL [BS97, MS99, MMZ+01] on our favourite CNF formula:

$$(p \lor \overline{u}) \land (q \lor r) \land (\overline{r} \lor w) \land (u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{p} \lor \overline{u})$$

- $p \stackrel{\mathrm{d}}{=} 0$
- $u \stackrel{p \lor \overline{u}}{=} 0$

decision level 1

- $q \stackrel{\mathrm{d}}{=} 0$
- $r \stackrel{q \lor r}{=} 1$
- $w \stackrel{\overline{r} \lor w}{=} 1$

decision level 2

- $x \stackrel{\mathrm{d}}{=} 0$
- $y \stackrel{u \lor x \lor y}{=} 1$
- $z \stackrel{x \lor \overline{y} \lor z}{=} 1$
- $\stackrel{\overline{y} \lor \overline{z}}{\bot}$

decision level 3

Decision

Free choice to assign value to variable

Notation $p \stackrel{\mathrm{d}}{=} 0$

Unit propagation

Forced choice to avoid falsifying clause

Given $p = 0$, clause $p \lor \overline{u}$ forces $u = 0$

Notation $u \stackrel{p \lor \overline{u}}{=} 0$ ($p \lor \overline{u}$ is reason clause)

Always propagate if possible, otherwise decide

Add to assignment trail

Continue until satisfying assignment or conflict

CONFLICT ANALYSIS

Time to analyse this conflict and learn from it!

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

$$\left.\begin{array}{l} \boxed{p \stackrel{\mathrm{d}}{=} 0} \\ u \stackrel{p \vee \overline{u}}{=} 0 \end{array}\right\} \text{decision level 1}$$

$$\left.\begin{array}{l} \boxed{q \stackrel{\mathrm{d}}{=} 0} \\ r \stackrel{q \vee r}{=} 1 \\ w \stackrel{\overline{r} \vee w}{=} 1 \end{array}\right\} \text{decision level 2}$$

$$\left.\begin{array}{l} \boxed{x \stackrel{\mathrm{d}}{=} 0} \\ y \stackrel{u \vee x \vee y}{=} 1 \\ z \stackrel{x \vee \overline{y} \vee z}{=} 1 \\ \stackrel{\overline{y} \vee \overline{z}}{\bot} \end{array}\right\} \text{decision level 3}$$

CONFLICT ANALYSIS

Time to analyse this conflict and learn from it!

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

$$\begin{array}{l}
\left.\begin{array}{c} p \stackrel{\mathrm{d}}{=} 0 \\ u \stackrel{p \vee \overline{u}}{=} 0 \end{array}\right\} \text{decision level 1} \\
\left.\begin{array}{c} q \stackrel{\mathrm{d}}{=} 0 \\ r \stackrel{q \vee r}{=} 1 \\ w \stackrel{\overline{r} \vee w}{=} 1 \end{array}\right\} \text{decision level 2} \\
\left.\begin{array}{c} x \stackrel{\mathrm{d}}{=} 0 \\ y \stackrel{u \vee x \vee y}{=} 1 \\ z \stackrel{x \vee \overline{y} \vee z}{=} 1 \\ \stackrel{\overline{y} \vee \overline{z}}{\bot} \end{array}\right\} \text{decision level 3}
\end{array}$$

Could backtrack by erasing conflict level & flipping last decision

CONFLICT ANALYSIS

Time to analyse this conflict and learn from it!

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

$$\left.\begin{array}{l} \boxed{p \overset{\mathrm{d}}{=} 0} \\ u \overset{p \vee \overline{u}}{=} 0 \end{array}\right\} \text{decision level 1}$$

$$\left.\begin{array}{l} \boxed{q \overset{\mathrm{d}}{=} 0} \\ r \overset{q \vee r}{=} 1 \\ w \overset{\overline{r} \vee w}{=} 1 \end{array}\right\} \text{decision level 2}$$

$$\left.\begin{array}{l} \boxed{x \overset{\mathrm{d}}{=} 0} \\ y \overset{u \vee x \vee y}{=} 1 \\ z \overset{x \vee \overline{y} \vee z}{=} 1 \\ \overset{\overline{y} \vee \overline{z}}{\perp} \end{array}\right\} \text{decision level 3}$$

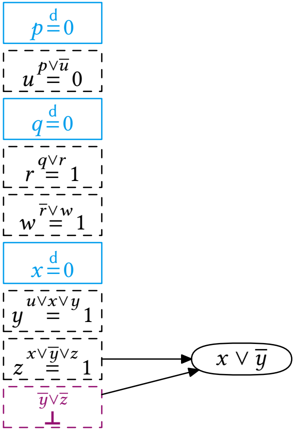
Could backtrack by erasing conflict level & flipping last decision

But want to learn from conflict and cut away as much of search space as possible

CONFLICT ANALYSIS

Time to analyse this conflict and learn from it!

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$



Could backtrack by erasing conflict level & flipping last decision

But want to learn from conflict and cut away as much of search space as possible

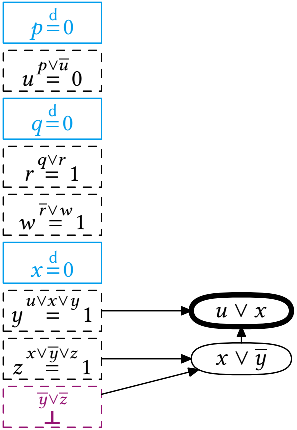
Case analysis over z for last two clauses:

- $x \vee \overline{y} \vee z$ wants $z = 1$
- $\overline{y} \vee \overline{z}$ wants $z = 0$
- Resolve clauses by merging them & removing z — must satisfy $x \vee \overline{y}$

CONFLICT ANALYSIS

Time to analyse this conflict and learn from it!

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$



Could backtrack by erasing conflict level & flipping last decision

But want to learn from conflict and cut away as much of search space as possible

Case analysis over z for last two clauses:

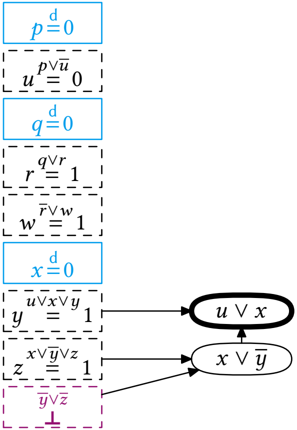
- $x \vee \overline{y} \vee z$ wants $z = 1$
- $\overline{y} \vee \overline{z}$ wants $z = 0$
- Resolve clauses by merging them & removing z — must satisfy $x \vee \overline{y}$

Repeat until UIP clause with only 1 variable at conflict level after last decision — learn and backjump

COMPLETE EXAMPLE OF CDCL EXECUTION

Backjump: undo max #decisions while learned clause propagates

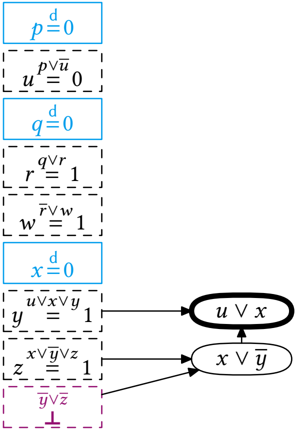
$$(p \lor \overline{u}) \land (q \lor r) \land (\overline{r} \lor w) \land (u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{p} \lor \overline{u})$$



COMPLETE EXAMPLE OF CDCL EXECUTION

Backjump: undo max #decisions while learned clause propagates

$$(p \lor \overline{u}) \land (q \lor r) \land (\overline{r} \lor w) \land (u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{p} \lor \overline{u})$$

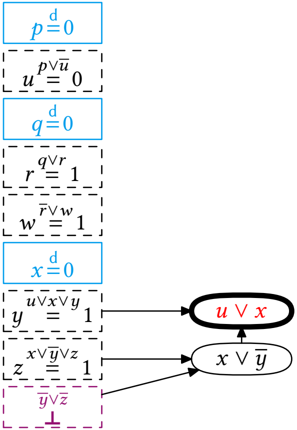


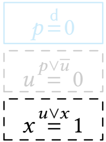
Assertion level 1 (2nd largest level in learned clause) — trim trail to that level

COMPLETE EXAMPLE OF CDCL EXECUTION

Backjump: undo max #decisions while learned clause propagates

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$





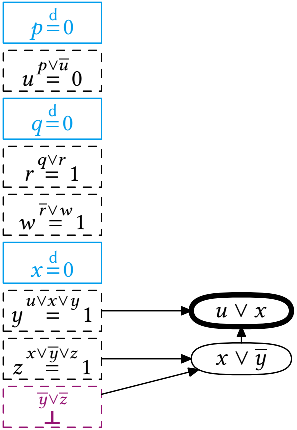
Assertion level 1 (2nd largest level in learned clause) — trim trail to that level

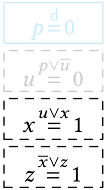
Now UIP literal guaranteed to flip (assert) — but this is a propagation, not a decision

COMPLETE EXAMPLE OF CDCL EXECUTION

Backjump: undo max #decisions while learned clause propagates

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$





Assertion level 1 (2nd largest level in learned clause) — trim trail to that level

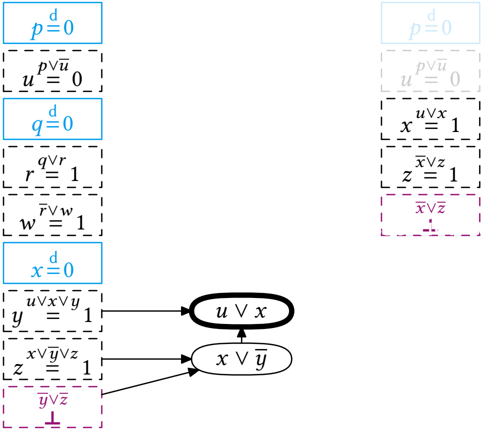
Now UIP literal guaranteed to flip (assert) — but this is a propagation, not a decision

Then continue as before…

COMPLETE EXAMPLE OF CDCL EXECUTION

Backjump: undo max #decisions while learned clause propagates

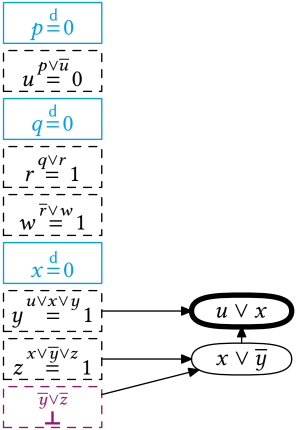
$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

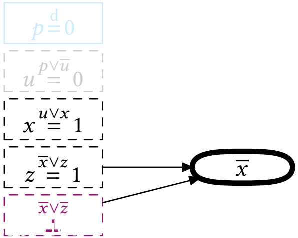


COMPLETE EXAMPLE OF CDCL EXECUTION

Backjump: undo max #decisions while learned clause propagates

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

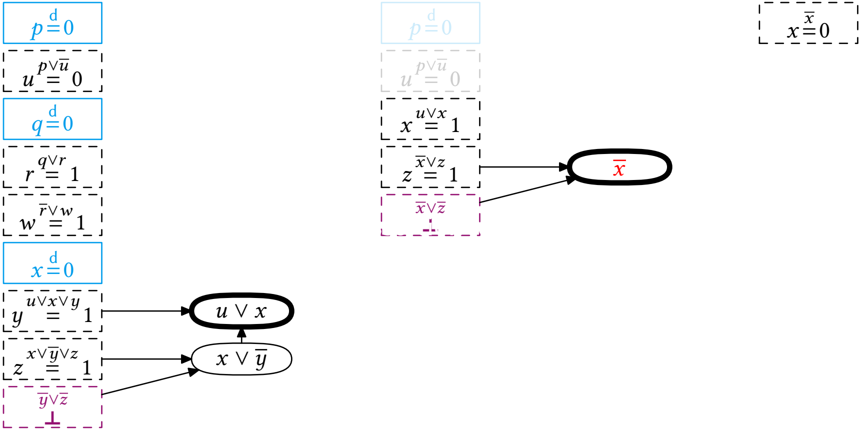




COMPLETE EXAMPLE OF CDCL EXECUTION

Backjump: undo max #decisions while learned clause propagates

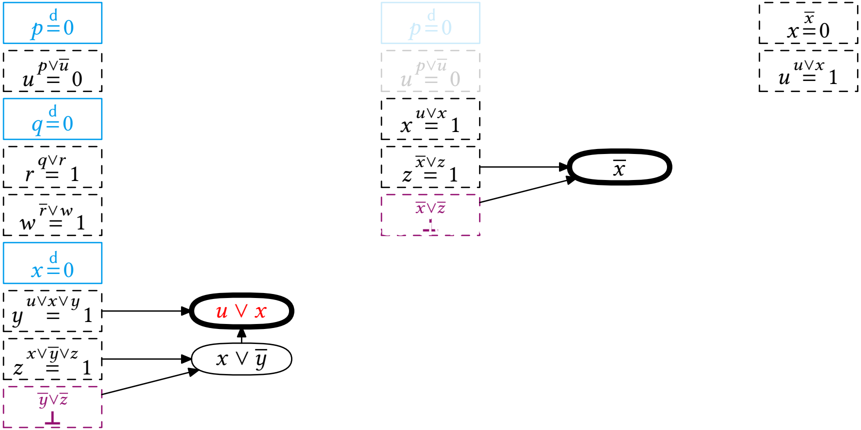
$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$



COMPLETE EXAMPLE OF CDCL EXECUTION

Backjump: undo max #decisions while learned clause propagates

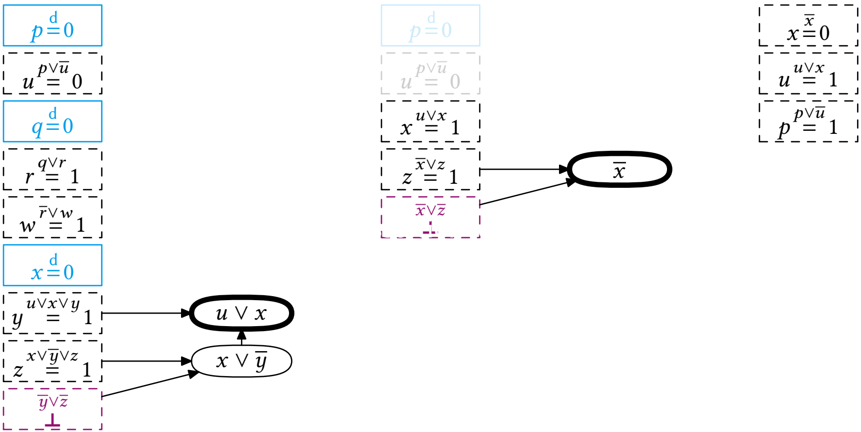
$$(p \lor \overline{u}) \land (q \lor r) \land (\overline{r} \lor w) \land (u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{p} \lor \overline{u})$$



COMPLETE EXAMPLE OF CDCL EXECUTION

Backjump: undo max #decisions while learned clause propagates

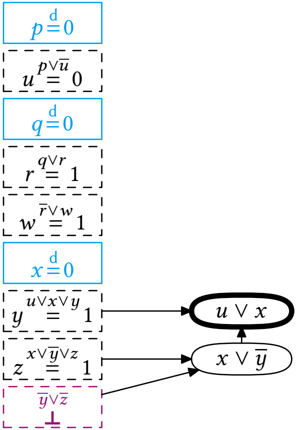
$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$

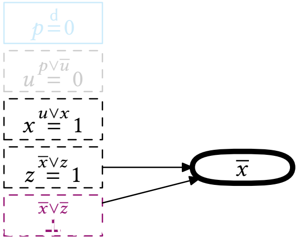


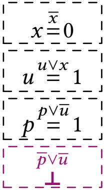
COMPLETE EXAMPLE OF CDCL EXECUTION

Backjump: undo max #decisions while learned clause propagates

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$



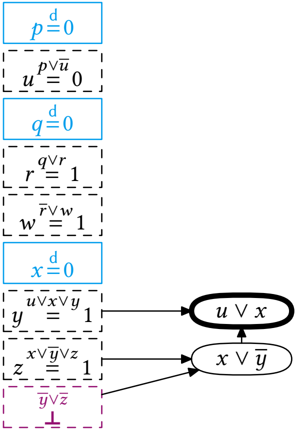


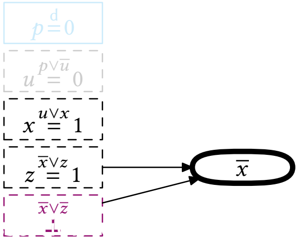


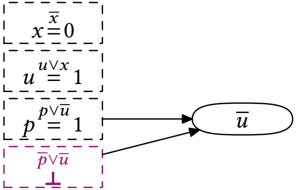
COMPLETE EXAMPLE OF CDCL EXECUTION

Backjump: undo max #decisions while learned clause propagates

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$



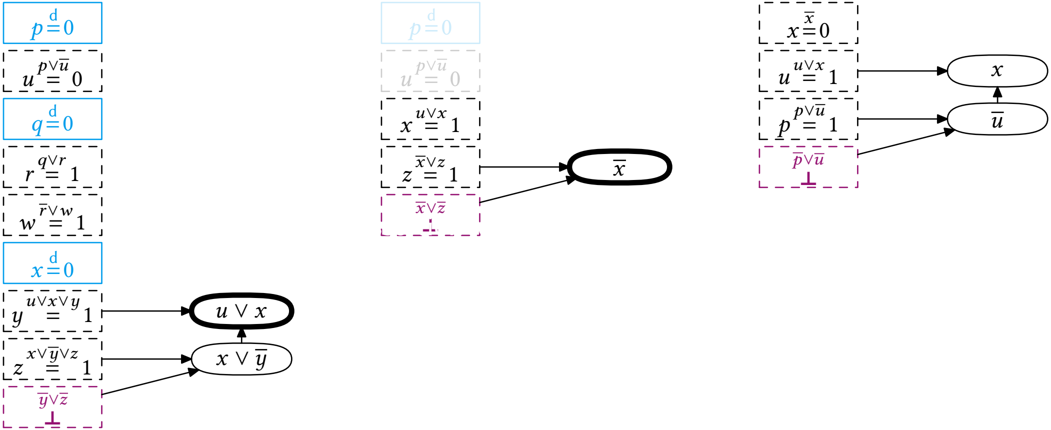




COMPLETE EXAMPLE OF CDCL EXECUTION

Backjump: undo max #decisions while learned clause propagates

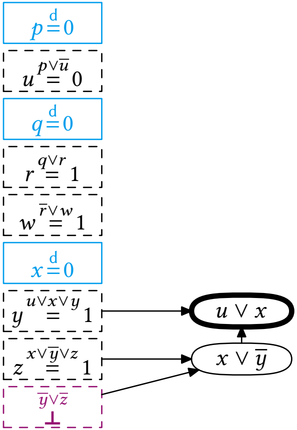
$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

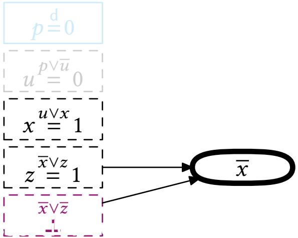


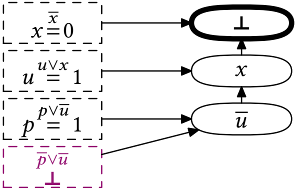
COMPLETE EXAMPLE OF CDCL EXECUTION

Backjump: undo max #decisions while learned clause propagates

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$







OUTLINE

CDCL REASONING AND THE RESOLUTION PROOF SYSTEM

To describe CDCL reasoning, need formal proof system for unsatisfiable formulas

CDCL REASONING AND THE RESOLUTION PROOF SYSTEM

To describe CDCL reasoning, need formal proof system for unsatisfiable formulas

Resolution proof system [Bla37, Rob65]

- Start with clauses of formula (axioms)
- Derive new clauses by resolution rule

$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D}$$

- Done when contradiction $\perp$ in form of empty clause derived

CDCL REASONING AND THE RESOLUTION PROOF SYSTEM

To describe CDCL reasoning, need formal proof system for unsatisfiable formulas

Resolution proof system [Bla37, Rob65]

- Start with clauses of formula (axioms)
- Derive new clauses by resolution rule

$$\frac{C \vee x \qquad D \vee \overline{x}}{C \vee D}$$

- Done when contradiction $\perp$ in form of empty clause derived

When run on unsatisfiable formula, CDCL generates resolution proof*

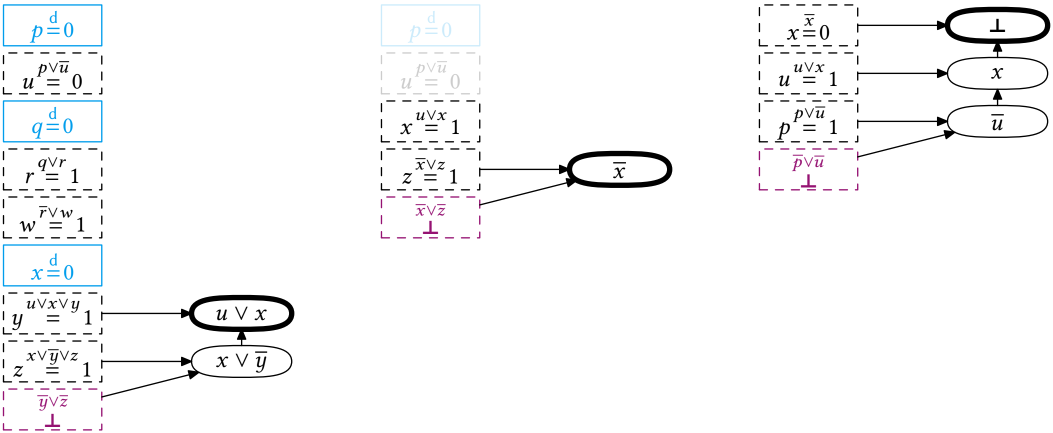
(*) Ignores pre- and inprocessing, but we will get there…

RESOLUTION PROOFS FROM CDCL EXECUTIONS

Obtain resolution proof…

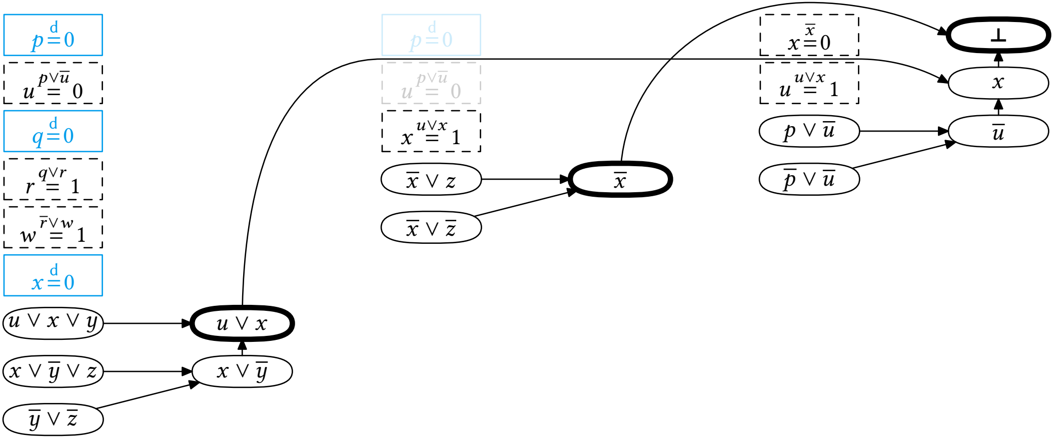
RESOLUTION PROOFS FROM CDCL EXECUTIONS

Obtain resolution proof from our example CDCL execution...



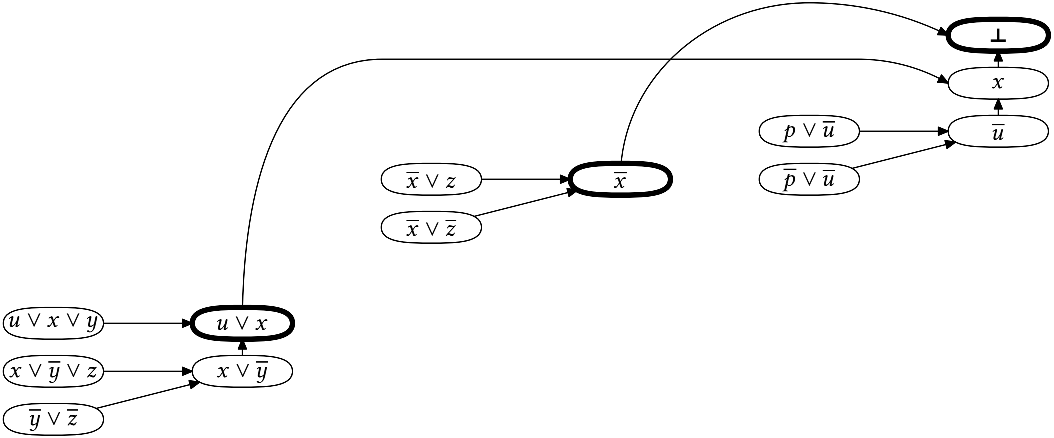
RESOLUTION PROOFS FROM CDCL EXECUTIONS

Obtain resolution proof from our example CDCL execution by stringing together conflict analyses:



RESOLUTION PROOFS FROM CDCL EXECUTIONS

Obtain resolution proof from our example CDCL execution by stringing together conflict analyses:



RUP PROOFS AND CDCL

But it turns out we can be lazier…

Fact

All learned clauses generated by CDCL solver are RUP clauses

RUP PROOFS AND CDCL

But it turns out we can be lazier…

Fact

All learned clauses generated by CDCL solver are RUP clauses

So shorter short proof of unsatisfiability for

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

is sequence of reverse unit propagation (RUP) clauses

1. $u \vee x$
2. $\overline{x}$
3. $\perp$

RUP PROOFS AND CDCL

But it turns out we can be lazier...

Fact

All learned clauses generated by CDCL solver are RUP clauses

So shorter short proof of unsatisfiability for

$$(p \lor \overline{u}) \land (q \lor r) \land (\overline{r} \lor w) \land (u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{p} \lor \overline{u})$$

is sequence of reverse unit propagation (RUP) clauses

1. $u \lor x$
2. $\overline{x}$
3. $\perp$

RUP PROOFS AND CDCL

But it turns out we can be lazier...

Fact

All learned clauses generated by CDCL solver are RUP clauses

So shorter short proof of unsatisfiability for

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

is sequence of reverse unit propagation (RUP) clauses

1. $u \vee x$
2. $\overline{x}$
3. $\perp$

RUP PROOFS AND CDCL

But it turns out we can be lazier...

Fact

All learned clauses generated by CDCL solver are RUP clauses

So shorter short proof of unsatisfiability for

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

is sequence of reverse unit propagation (RUP) clauses

1. $u \vee x$
2. $\overline{x}$
3. $\perp$

RUP PROOFS AND CDCL

But it turns out we can be lazier…

Fact

All learned clauses generated by CDCL solver are RUP clauses

So shorter short proof of unsatisfiability for

$$(p \lor \overline{u}) \land (q \lor r) \land (\overline{r} \lor w) \land (u \lor x \lor y) \land (x \lor \overline{y} \lor z) \land (\overline{x} \lor z) \land (\overline{y} \lor \overline{z}) \land (\overline{x} \lor \overline{z}) \land (\overline{p} \lor \overline{u})$$

is sequence of reverse unit propagation (RUP) clauses

1. $u \lor x$
2. $\overline{x}$
3. $\perp$

RUP PROOFS AND CDCL

But it turns out we can be lazier…

Fact

All learned clauses generated by CDCL solver are RUP clauses

So shorter short proof of unsatisfiability for

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

is sequence of reverse unit propagation (RUP) clauses

1. $u \vee x$
2. $\overline{x}$
3. $\perp$

RUP PROOFS AND CDCL

But it turns out we can be lazier…

Fact

All learned clauses generated by CDCL solver are RUP clauses

So shorter short proof of unsatisfiability for

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

is sequence of reverse unit propagation (RUP) clauses

1. $u \vee x$
2. $\overline{x}$
3. $\perp$

RUP PROOFS AND CDCL

But it turns out we can be lazier…

Fact

All learned clauses generated by CDCL solver are RUP clauses

So shorter short proof of unsatisfiability for

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

is sequence of reverse unit propagation (RUP) clauses

1. $u \vee x$
2. $\overline{x}$
3. $\perp$

RUP PROOFS AND CDCL

But it turns out we can be lazier...

Fact

All learned clauses generated by CDCL solver are RUP clauses

So shorter short proof of unsatisfiability for

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

is sequence of reverse unit propagation (RUP) clauses

1. $u \vee x$
2. $\overline{x}$
3. $\perp$

RUP PROOFS AND CDCL

But it turns out we can be lazier…

Fact

All learned clauses generated by CDCL solver are RUP clauses

So shorter short proof of unsatisfiability for

$$(p \vee \overline{u}) \wedge (q \vee r) \wedge (\overline{r} \vee w) \wedge (u \vee x \vee y) \wedge (x \vee \overline{y} \vee z) \wedge (\overline{x} \vee z) \wedge (\overline{y} \vee \overline{z}) \wedge (\overline{x} \vee \overline{z}) \wedge (\overline{p} \vee \overline{u})$$

is sequence of reverse unit propagation (RUP) clauses

1. $u \vee x$
2. $\overline{x}$
3. $\perp$

MORE INGREDIENTS IN PROOF LOGGING FOR SAT

Fact

RUP proofs can be viewed as shorthand for resolution proofs

See [BN21] for more on this and connections to SAT solving

But RUP and resolution are not enough for preprocessing, inprocessing, and some other kinds of reasoning

EXTENSION VARIABLES, PART 1

Suppose we want a variable a encoding

$$a \Leftrightarrow (x \land y)$$

Extended resolution [Tse68]

Resolution rule plus extension rule introducing clauses

$$a \lor \overline{x} \lor \overline{y} \qquad \overline{a} \lor x \qquad \overline{a} \lor y$$

for fresh variable a (this is fine since a doesn't appear anywhere previously)

EXTENSION VARIABLES, PART 1

Suppose we want a variable a encoding

$$a \Leftrightarrow (x \wedge y)$$

Extended resolution [Tse68]

Resolution rule plus extension rule introducing clauses

$$a \vee \overline{x} \vee \overline{y} \qquad \overline{a} \vee x \qquad \overline{a} \vee y$$

for fresh variable a (this is fine since a doesn't appear anywhere previously)

Fact

Extended resolution (RUP + definition of new variables) is essentially equivalent to the DRAT proof logging system most commonly used for SAT solving

OUTLINE

1. Introduction
    1. The Success of Combinatorial Solving (and the Dirty Little Secret…)
    2. Ensuring Correctness with the Help of Proof Logging
2. Proof Logging for SAT
    1. SAT Basics
    2. DPLL and CDCL
    3. Proof System for SAT Proof Logging
3. Pseudo-Boolean Proof Logging
    1. Pseudo-Boolean Constraints and Cutting Planes Reasoning
    2. Pseudo-Boolean Proof Logging for SAT Solving
    3. More Pseudo-Boolean Proof Logging Rules
4. Proof Logging for SAT-Based Optimisation (MaxSAT solving)
    1. Linear SAT-UNSAT Search
    2. Core-Guided Search
    3. Branch-And-Bound Search
5. Conclusion

WHY AREN'T WE DONE?

Practical limitations of current SAT proof logging technology:

- Difficulties dealing with stronger reasoning efficiently (even for SAT solving)
- Clausal proofs can't easily reflect what algorithms for other problems do

WHY AREN'T WE DONE?

Practical limitations of current SAT proof logging technology:

- Difficulties dealing with stronger reasoning efficiently (even for SAT solving)
- Clausal proofs can't easily reflect what algorithms for other problems do

Surprising claim: a slight change to 0-1 integer linear inequalities does the job!

- Enables proof logging for advanced SAT techniques so far beyond reach for efficient DRAT proof logging:
  - Cardinality reasoning
  - Gaussian elimination
  - Symmetry breaking
- Supports use of SAT solvers for optimisation problems (MaxSAT)
- Can justify graph reasoning without knowing what a graph is
- Can justify constraint programming inference without knowing what an integer variable is

OUTLINE

PSEUDO-BOOLEAN CONSTRAINTS

0–1 integer linear inequalities or (linear) pseudo-Boolean constraints:

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

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

PSEUDO-BOOLEAN CONSTRAINTS

0–1 integer linear inequalities or (linear) pseudo-Boolean constraints:

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

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

Sometimes convenient to use normalized form [Bar95] with all a_i, A positive (without loss of generality)

SOME TYPES OF PSEUDO-BOOLEAN CONSTRAINTS

1. Clauses

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

2. Cardinality constraints

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

3. General pseudo-Boolean constraints

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

PSEUDO-BOOLEAN REASONING: CUTTING PLANES

Input/model axioms From the input

PSEUDO-BOOLEAN REASONING: CUTTING PLANES

Input/model axioms — From the input

Literal axioms — $$\frac{}{\ell_i \geq 0}$$

PSEUDO-BOOLEAN REASONING: CUTTING PLANES

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

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

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

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

CUTTING PLANES TOY EXAMPLE

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

CUTTING PLANES TOY EXAMPLE

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

CUTTING PLANES TOY EXAMPLE

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

CUTTING PLANES TOY EXAMPLE

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

CUTTING PLANES TOY EXAMPLE

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

CUTTING PLANES TOY EXAMPLE

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

CUTTING PLANES TOY EXAMPLE

$$\dfrac{\dfrac{\dfrac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4}\ \text{Multiply by 2} \qquad w + 2x + 4y + 2z \geq 5}{3w + 6x + 6y + 2z \geq 9}\ \text{Add} \qquad \dfrac{\overline{z} \geq 0}{2\overline{z} \geq 0}\ \text{Multiply by 2}}{3w + 6x + 6y \geq 7}\ \text{Add}$$

CUTTING PLANES TOY EXAMPLE

$$
\dfrac{
  \dfrac{
    \text{Multiply by 2}\;\dfrac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4}
    \qquad
    w + 2x + 4y + 2z \geq 5
  }{3w + 6x + 6y + 2z \geq 9}\;\text{Add}
  \qquad
  \dfrac{\overline{z} \geq 0}{2\overline{z} \geq 0}\;\text{Multiply by 2}
}{
  \text{Divide by 3}\;\dfrac{3w + 6x + 6y \geq 7}{w + 2x + 2y \geq 2\frac{1}{3}}
}\;\text{Add}
$$

CUTTING PLANES TOY EXAMPLE

$$
\dfrac{\dfrac{\dfrac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4}\text{ Multiply by 2} \qquad w + 2x + 4y + 2z \geq 5}{3w + 6x + 6y + 2z \geq 9}\text{ Add} \qquad \dfrac{\overline{z} \geq 0}{2\overline{z} \geq 0}\text{ Multiply by 2}}{\dfrac{3w + 6x + 6y \geq 7}{w + 2x + 2y \geq 3}\text{ Divide by 3}}\text{ Add}
$$

CUTTING PLANES TOY EXAMPLE

$$\text{Divide by 3}\ \dfrac{\text{Add}\ \dfrac{\text{Add}\ \dfrac{\text{Multiply by 2}\ \dfrac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4} \qquad w + 2x + 4y + 2z \geq 5}{3w + 6x + 6y + 2z \geq 9} \qquad \dfrac{\overline{z} \geq 0}{2\overline{z} \geq 0}\ \text{Multiply by 2}}{3w + 6x + 6y \geq 7}}{w + 2x + 2y \geq 3}$$

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

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

CUTTING PLANES TOY EXAMPLE

$$\dfrac{\dfrac{\dfrac{\dfrac{w + 2x + y \geq 2}{2w + 4x + 2y \geq 4}\,\text{Multiply by 2} \qquad w + 2x + 4y + 2z \geq 5}{3w + 6x + 6y + 2z \geq 9}\,\text{Add} \qquad \dfrac{\overline{z} \geq 0}{2\overline{z} \geq 0}\,\text{Multiply by 2}}{3w + 6x + 6y \geq 7}\,\text{Add}}{w + 2x + 2y \geq 3}\,\text{Divide by 3}$$

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

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

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

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

OUTLINE

1. Introduction
   1. The Success of Combinatorial Solving (and the Dirty Little Secret...)
   2. Ensuring Correctness with the Help of Proof Logging
2. Proof Logging for SAT
   1. SAT Basics
   2. DPLL and CDCL
   3. Proof System for SAT Proof Logging
3. Pseudo-Boolean Proof Logging
   1. Pseudo-Boolean Constraints and Cutting Planes Reasoning
   2. Pseudo-Boolean Proof Logging for SAT Solving
   3. More Pseudo-Boolean Proof Logging Rules
4. Proof Logging for SAT-Based Optimisation (MaxSAT solving)
   1. Linear SAT-UNSAT Search
   2. Core-Guided Search
   3. Branch-And-Bound Search
5. Conclusion

RESOLUTION AND CUTTING PLANES

To simulate resolution step such as

$$\frac{\overline{y} \vee \overline{z} \qquad x \vee \overline{y} \vee z}{x \vee \overline{y}}$$

we can perform the cutting planes steps

$$\text{Divide by 2}\ \dfrac{\text{Add}\ \dfrac{\overline{y} + \overline{z} \geq 1 \qquad x + \overline{y} + z \geq 1}{x + 2\overline{y} \geq 1}}{x + \overline{y} \geq 1}$$

RESOLUTION AND CUTTING PLANES

To simulate resolution step such as

$$\frac{\overline{y} \vee \overline{z} \qquad x \vee \overline{y} \vee z}{x \vee \overline{y}}$$

we can perform the cutting planes steps

$$\text{Add}\ \dfrac{\overline{y} + \overline{z} \geq 1 \qquad x + \overline{y} + z \geq 1}{\text{Divide by 2}\ \dfrac{x + 2\overline{y} \geq 1}{x + \overline{y} \geq 1}}$$

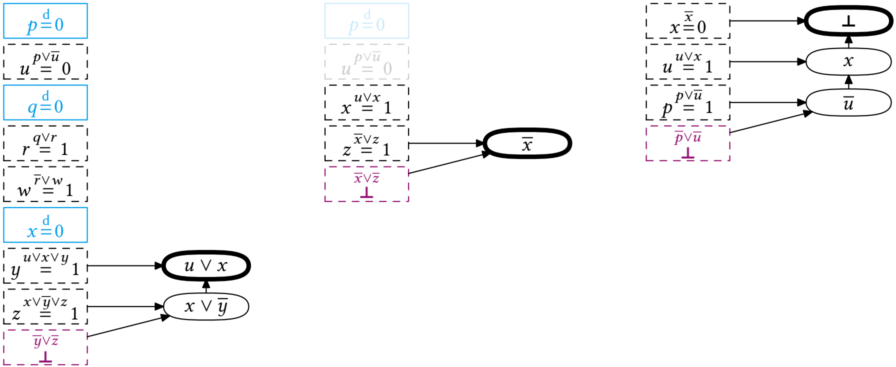
Given that the premises are clauses 7 and 5 in our example CNF formula, using references

$$\text{Constraint 7} \doteq \overline{y} + \overline{z} \geq 1$$
$$\text{Constraint 5} \doteq x + \overline{y} + z \geq 1$$

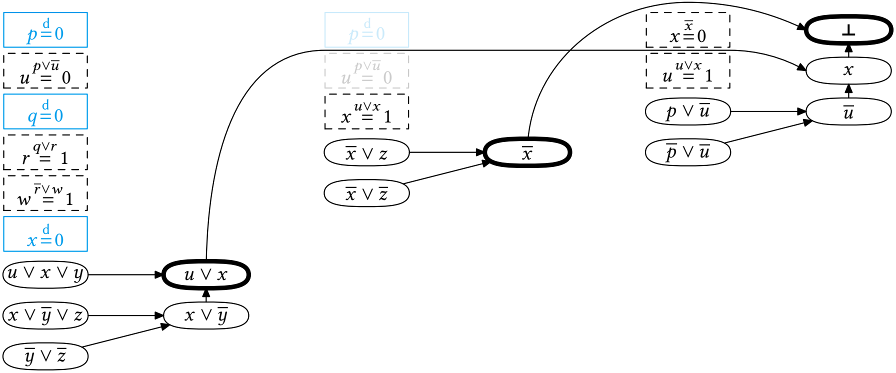
we can write this in the proof log as

```
pol 7 5 + 2 d
```

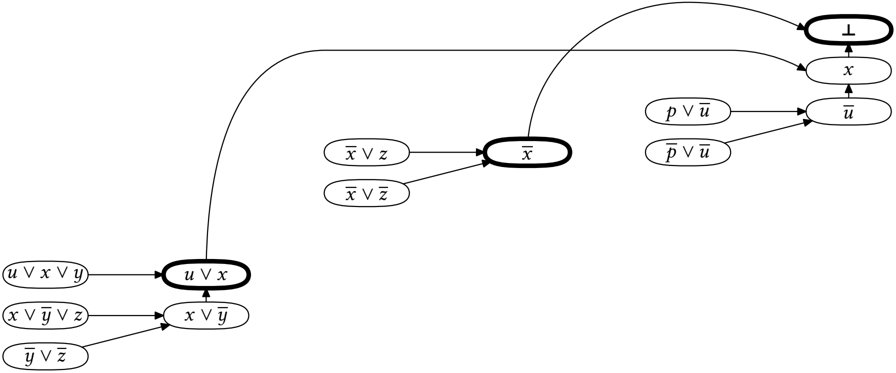
PSEUDO-BOOLEAN PROOF LOGGING FOR CDCL EXAMPLE



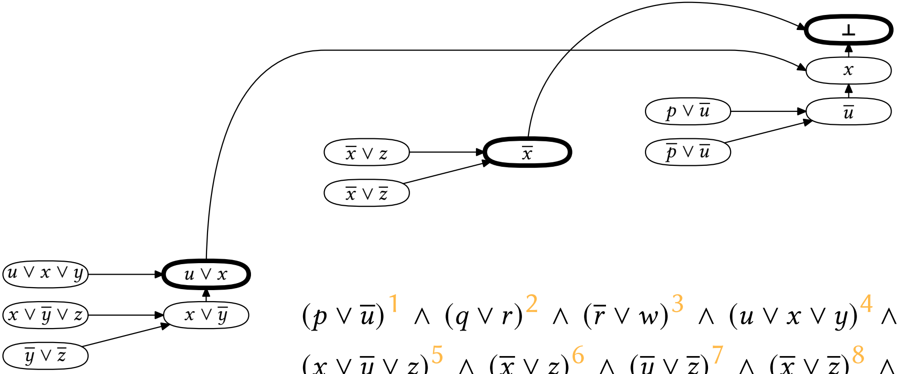
PSEUDO-BOOLEAN PROOF LOGGING FOR CDCL EXAMPLE



PSEUDO-BOOLEAN PROOF LOGGING FOR CDCL EXAMPLE

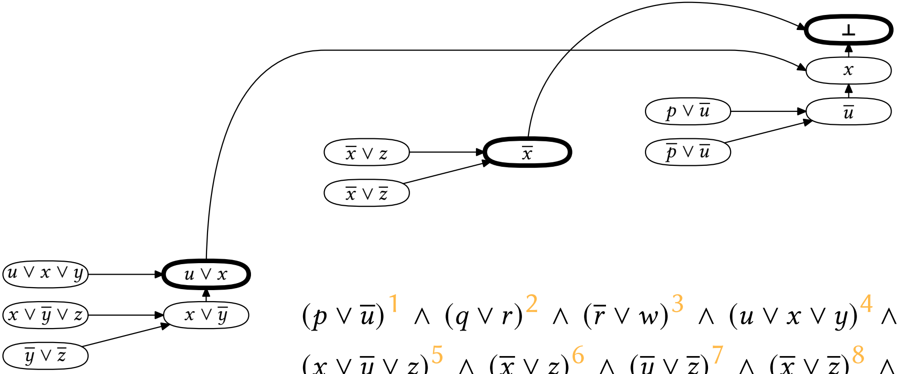


PSEUDO-BOOLEAN PROOF LOGGING FOR CDCL EXAMPLE



$$(p \vee \overline{u})^{1} \wedge (q \vee r)^{2} \wedge (\overline{r} \vee w)^{3} \wedge (u \vee x \vee y)^{4} \wedge$$
$$(x \vee \overline{y} \vee z)^{5} \wedge (\overline{x} \vee z)^{6} \wedge (\overline{y} \vee \overline{z})^{7} \wedge (\overline{x} \vee \overline{z})^{8} \wedge (\overline{p} \vee \overline{u})^{9}$$

PSEUDO-BOOLEAN PROOF LOGGING FOR CDCL EXAMPLE



$$(p \vee \overline{u})^{1} \wedge (q \vee r)^{2} \wedge (\overline{r} \vee w)^{3} \wedge (u \vee x \vee y)^{4} \wedge$$
$$(x \vee \overline{y} \vee z)^{5} \wedge (\overline{x} \vee z)^{6} \wedge (\overline{y} \vee \overline{z})^{7} \wedge (\overline{x} \vee \overline{z})^{8} \wedge (\overline{p} \vee \overline{u})^{9}$$

`pol 7 5 + 2 d 4 + 2 d` $\rightsquigarrow$ Constraint 10 $\doteq u + x \geq 1$

`pol 8 6 + 2 d` $\rightsquigarrow$ Constraint 11 $\doteq \overline{x} \geq 1$

`pol 9 1 + 2 d 10 + 2 d 11 + 2 d` $\rightsquigarrow$ Constraint 12 $\doteq 0 \geq 1$ ↯

RUP REVISITED

Can define (reverse) unit propagation in a pseudo-Boolean setting

Constraint C propagates variable x if setting x to "wrong value" would make C unsatisfiable

E.g., if x_5 is false,

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

would propagate $\overline{x}_4$ (since other coefficients do not add up to 7)

RUP REVISITED

Can define (reverse) unit propagation in a pseudo-Boolean setting

Constraint C propagates variable x if setting x to "wrong value" would make C unsatisfiable

E.g., if x_5 is false,

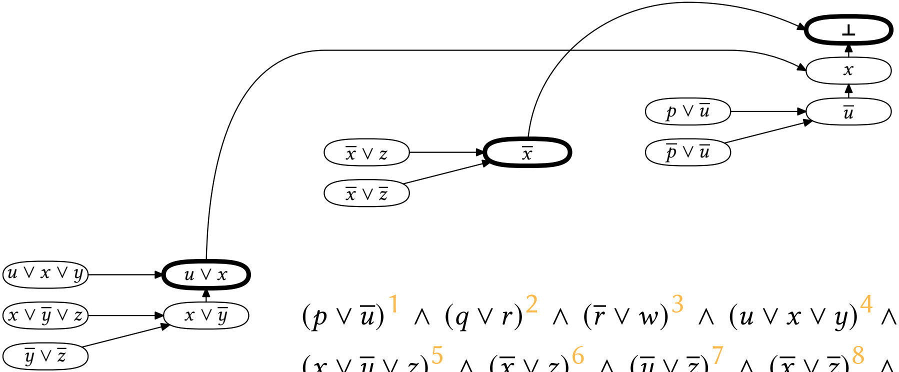
$$x_1 + 2\overline{x}_2 + 3x_3 + 4\overline{x}_4 + 5x_5 \geq 7$$

would propagate $\overline{x}_4$ (since other coefficients do not add up to 7)

Risk for confusion:

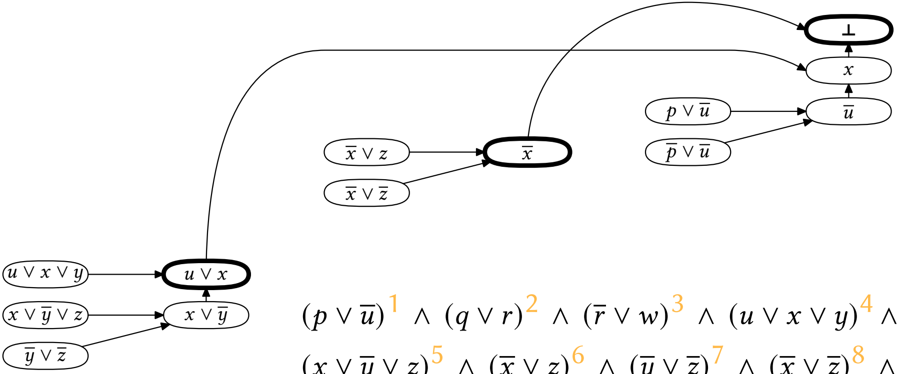
- Constraint programming people might call this (reverse) integer bounds consistency
  - Does the same thing if we're working with clauses
  - More interesting for general pseudo-Boolean constraints
- SAT people beware: constraints can propagate multiple times and multiple variables

PB PROOF LOGGING FOR EXAMPLE CDCL EXECUTION WITH RUP



$$(p \vee \overline{u})^{1} \wedge (q \vee r)^{2} \wedge (\overline{r} \vee w)^{3} \wedge (u \vee x \vee y)^{4} \wedge$$
$$(x \vee \overline{y} \vee z)^{5} \wedge (\overline{x} \vee z)^{6} \wedge (\overline{y} \vee \overline{z})^{7} \wedge (\overline{x} \vee \overline{z})^{8} \wedge (\overline{p} \vee \overline{u})^{9}$$

PB PROOF LOGGING FOR EXAMPLE CDCL EXECUTION WITH RUP



$$(p \vee \overline{u})^{1} \wedge (q \vee r)^{2} \wedge (\overline{r} \vee w)^{3} \wedge (u \vee x \vee y)^{4} \wedge$$
$$(x \vee \overline{y} \vee z)^{5} \wedge (\overline{x} \vee z)^{6} \wedge (\overline{y} \vee \overline{z})^{7} \wedge (\overline{x} \vee \overline{z})^{8} \wedge (\overline{p} \vee \overline{u})^{9}$$

```
rup 1 u 1 x >= 1 ;
rup 1 ~x >= 1 ;
rup >= 1 ;
```

$\rightsquigarrow$ Constraint 10 $\doteq u + x \geq 1$

$\rightsquigarrow$ Constraint 11 $\doteq \overline{x} \geq 1$

$\rightsquigarrow$ Constraint 12 $\doteq 0 \geq 1$ ↯

OUTLINE

EXTENSION VARIABLES, PART 2

Suppose we want new, fresh variable a encoding

$$a \Leftrightarrow (3x + 2y + z + w \geq 3)$$

This time, introduce constraints

$$3\overline{a} + 3x + 2y + z + w \geq 3 \qquad 5a + 3\overline{x} + 2\overline{y} + \overline{z} + \overline{w} \geq 5$$

Again, needs support from the proof system

PROOF LOGS FOR “EXTENDED CUTTING PLANES”

For satisfiable instances: just specify a satisfying assignment.

For unsatisfiability: a sequence of pseudo-Boolean constraints in (slight extension of) OPB format [RM16]

- Each constraint follows “obviously” from what is known so far
- Either implicitly, by RUP…
- Or by an explicit cutting planes derivation…
- Or as an extension variable reifying a new constraint*
- Final constraint is $0 \geq 1$

PROOF LOGS FOR “EXTENDED CUTTING PLANES”

For satisfiable instances: just specify a satisfying assignment.

For unsatisfiability: a sequence of pseudo-Boolean constraints in (slight extension of) OPB format [RM16]

- Each constraint follows “obviously” from what is known so far
- Either implicitly, by RUP…
- Or by an explicit cutting planes derivation…
- Or as an extension variable reifying a new constraint*
- Final constraint is $0 \geq 1$

(*) Not actually implemented this way — details to come later…

DELETING CONSTRAINTS

In practice, important to erase constraints to save memory and time during verification

Fairly straightforward to deal with from the point of view of proof logging

So ignored in this tutorial for simplicity and clarity

ENUMERATION AND OPTIMISATION PROBLEMS

Enumeration:

- When a solution is found, can log it
- Introduces a new constraint saying “not this solution”
- So the proof semantics is “infeasible, except for all the solutions I told you about”

ENUMERATION AND OPTIMISATION PROBLEMS

Enumeration:

- When a solution is found, can log it
- Introduces a new constraint saying "not this solution"
- So the proof semantics is "infeasible, except for all the solutions I told you about"

For optimisation:

- Define an objective $f = \sum_i w_i \ell_i$, $w_i \in \mathbb{Z}$, to minimise subject to the contraints in the formula
- To maximise, negate objective
- Log a solution α; get an objective-improving constraint $\sum_i w_i \ell_i \leq -1 + \sum_i w_i \alpha(\ell_i)$
- Semantics for proof of optimality: "infeasible to find better solution than best so far"

PSEUDO-BOOLEAN PROOF LOGGING — HOW AND WHY?

If problem is (special case of) 0–1 integer linear program (ILP)

- just do proof logging

PSEUDO-BOOLEAN PROOF LOGGING — HOW AND WHY?

If problem is (special case of) 0–1 integer linear program (ILP)

- just do proof logging

Otherwise

- do trusted or verified translation to 0–1 ILP
- provide proof logging for 0–1 ILP formulation

PSEUDO-BOOLEAN PROOF LOGGING — HOW AND WHY?

If problem is (special case of) 0–1 integer linear program (ILP)

- just do proof logging

Otherwise

- do trusted or verified translation to 0–1 ILP
- provide proof logging for 0–1 ILP formulation

Proof logging philosophy:

- **do not change** input for solver
- **do not change** reasoning in solver
- **only** add print statements (in PB format) here and there

PSEUDO-BOOLEAN PROOF LOGGING — HOW AND WHY?

If problem is (special case of) 0–1 integer linear program (ILP)

- just do proof logging

Otherwise

- do trusted or verified translation to 0–1 ILP
- provide proof logging for 0–1 ILP formulation

Proof logging philosophy:

- **do not change** input for solver
- **do not change** reasoning in solver
- **only** add print statements (in PB format) here and there

Goldilocks compromise between expressivity and simplicity:

1. 0–1 ILP expressive formalism for combinatorial problems (including objective)
2. Powerful reasoning capturing many combinatorial arguments (even for SAT)
3. Efficient reification of constraints

PSEUDO-BOOLEAN PROOF LOGGING — HOW AND WHY?

If problem is (special case of) 0–1 integer linear program (ILP)

- just do proof logging

Otherwise

- do trusted or verified translation to 0–1 ILP
- provide proof logging for 0–1 ILP formulation

Proof logging philosophy:

- **do not change** input for solver
- **do not change** reasoning in solver
- **only** add print statements (in PB format) here and there

Goldilocks compromise between expressivity and simplicity:

1. 0–1 ILP expressive formalism for combinatorial problems (including objective)
2. Powerful reasoning capturing many combinatorial arguments (even for SAT)
3. Efficient reification of constraints — example:

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

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

PSEUDO-BOOLEAN PROOF LOGGING — HOW AND WHY?

If problem is (special case of) 0–1 integer linear program (ILP)

- just do proof logging

Otherwise

- do trusted or verified translation to 0–1 ILP
- provide proof logging for 0–1 ILP formulation

Proof logging philosophy:

- **do not change** input for solver
- **do not change** reasoning in solver
- **only** add print statements (in PB format) here and there

Goldilocks compromise between expressivity and simplicity:

1. 0–1 ILP expressive formalism for combinatorial problems (including objective)
2. Powerful reasoning capturing many combinatorial arguments (even for SAT)
3. Efficient reification of constraints — example:

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

$$r \Leftarrow x_1 + 2\overline{x}_2 + 3x_3 + 4\overline{x}_4 + 5x_5 \geq 7 \qquad 9r + \overline{x}_1 + 2x_2 + 3\overline{x}_3 + 4x_4 + 5\overline{x}_5 \geq 9$$

THE VERIPB FORMAT AND TOOL

`https://gitlab.com/MIAOresearch/software/VeriPB`

Released under MIT Licence

Various features to help development:

- Extended variable name syntax allowing human-readable names
- Proof tracing
- "Trust me" assertions for incremental proof logging

Documentation:

- Description of VeriPB checker [BMM+23] used in SAT 2023 competition (`https://satcompetition.github.io/2023/checkers.html`)
- Specific details on different proof logging techniques covered in research papers [EGMN20, GMN20, GMM+20, GN21, GMN22, GMNO22, VDB22, BBN+23, BGMN23, MM23]
- Lots of concrete example files at `https://gitlab.com/MIAOresearch/software/VeriPB`

OUTLINE

CERTIFIED MAXIMUM SATISFIABILITY (MAXSAT) SOLVING

Minimize linear objective subject to satisfying formula in conjunctive normal form (CNF)

$$\begin{aligned} \min\ & 2x_1 + x_2 \\ \text{s.t. } & x_1 \vee \overline{z} \\ & z \vee x_2 \end{aligned} \longrightarrow \boxed{\text{MaxSAT solver}} \longrightarrow$$

Result:
optimum 1

Many MaxSAT solvers internally make use of SAT solver.

CERTIFIED MAXIMUM SATISFIABILITY (MAXSAT) SOLVING

Minimize linear objective subject to satisfying formula in conjunctive normal form (CNF)

$$\begin{aligned} \min\ & 2x_1 + x_2 \\ \text{s.t. } & x_1 \vee \overline{z} \\ & z \vee x_2 \end{aligned} \longrightarrow \boxed{\text{MaxSAT solver}} \longrightarrow \text{Result: optimum } 1$$

Many MaxSAT solvers internally make use of SAT solver. Idea:

- Find optimal solution (checking that it *is* a solution is easy)
- Add clauses claiming a better solution exists
- Use one extra SAT call to get proof of optimality (with standard SAT proof logging)

CERTIFIED MAXIMUM SATISFIABILITY (MAXSAT) SOLVING

Minimize linear objective subject to satisfying formula in conjunctive normal form (CNF)

$$\begin{aligned} \min\ & 2x_1 + x_2 \\ \text{s.t. } & x_1 \vee \overline{z} \\ & z \vee x_2 \end{aligned} \longrightarrow \boxed{\text{MaxSAT solver}} \longrightarrow \text{Result: optimum } 1$$

Many MaxSAT solvers internally make use of SAT solver. Idea:

- Find optimal solution (checking that it *is* a solution is easy)
- Add clauses claiming a better solution exists
- Use one extra SAT call to get proof of optimality (with standard SAT proof logging)

Does not work

CERTIFIED MAXIMUM SATISFIABILITY (MAXSAT) SOLVING

Minimize linear objective subject to satisfying formula in conjunctive normal form (CNF)

$$\begin{aligned} \min\ & 2x_1 + x_2 \\ \text{s.t. } & x_1 \vee \overline{z} \\ & z \vee x_2 \end{aligned} \longrightarrow \boxed{\text{MaxSAT solver}} \longrightarrow \text{Result: optimum } 1$$

Many MaxSAT solvers internally make use of SAT solver. Idea:

- Find optimal solution (checking that it *is* a solution is easy)
- Add clauses claiming a better solution exists
  Requires proof logging — can be done with VeriPB
- Use one extra SAT call to get proof of optimality (with standard SAT proof logging)
  Causes serious overhead

Does not work

CERTIFIED MAXIMUM SATISFIABILITY (MAXSAT) SOLVING

Minimize linear objective subject to satisfying formula in conjunctive normal form (CNF)

$$\begin{aligned} \min\ & 2x_1 + x_2 \\ \text{s.t. } & x_1 \lor \overline{z} \\ & z \lor x_2 \end{aligned}$$

→ MaxSAT solver → Result: optimum 1

Many MaxSAT solvers internally make use of SAT solver. Idea:

- Find optimal solution (checking that it *is* a solution is easy)
- Add clauses claiming a better solution exists
  Requires proof logging — can be done with VeriPB
- Use one extra SAT call to get proof of optimality (with standard SAT proof logging)
  Causes serious overhead

Does not work Only proves answer correct, not reasoning within solver!

MAXSAT SOLVERS

Main categories:

- Linear SAT-UNSAT search (proof logging [VDB22, Van23, BBN+24])
  1. Call SAT solver to find some solution
  2. Add clauses encoding "I want a better solution"
  3. Repeat (last found solution is optimal)

MAXSAT SOLVERS

Main categories:

- Linear SAT-UNSAT search (proof logging [VDB22, Van23, BBN$^+$24])
  1. Call SAT solver to find some solution
  2. Add clauses encoding "I want a better solution"
  3. Repeat (last found solution is optimal)
- Core-guided search (proof logging [BBN$^+$23])
  1. Call SAT solver to find solution under most optimistic assumptions
  2. If impossible, rewrite objective given output of SAT solver
  3. Repeat (first solution is optimal)

MAXSAT SOLVERS

Main categories:

- Linear SAT-UNSAT search (proof logging [VDB22, Van23, BBN+24])
  1. Call SAT solver to find some solution
  2. Add clauses encoding "I want a better solution"
  3. Repeat (last found solution is optimal)
- Core-guided search (proof logging [BBN+23])
  1. Call SAT solver to find solution under most optimistic assumptions
  2. If impossible, rewrite objective given output of SAT solver
  3. Repeat (first solution is optimal)
- Branch-and-bound search (proof logging coming soon)
  1. Run CDCL SAT solver
  2. While running, add bounding constraints

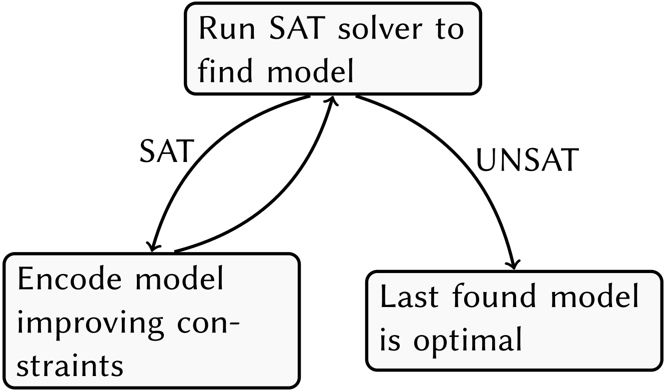
MAXSAT SOLVERS

Main categories:

- Linear SAT-UNSAT search (proof logging [VDB22, Van23, BBN+24])
  1. Call SAT solver to find some solution
  2. Add clauses encoding "I want a better solution"
  3. Repeat (last found solution is optimal)
- Core-guided search (proof logging [BBN+23])
  1. Call SAT solver to find solution under most optimistic assumptions
  2. If impossible, rewrite objective given output of SAT solver
  3. Repeat (first solution is optimal)
- Branch-and-bound search (proof logging coming soon)
  1. Run CDCL SAT solver
  2. While running, add bounding constraints
- Implicit Hitting Set (No proof logging available yet)
  1. Call SAT solver to find solution under most optimistic assumptions
  2. Use hitting set solver (MIP solver) to recompute what most possible optimistic assumptions are
  3. Repeat (first solution is optimal)

OUTLINE

LINEAR SAT-UNSAT SEARCH



CERTIFIED LSU SEARCH (EXAMPLE)

$\overline{x}_1 \vee x_2$ $\quad$ $\overline{x}_1 \vee \overline{x}_2 \vee r_1$

$x_1 \vee \overline{x}_2$ $\quad$ $x_1 \vee x_2 \vee r_2$

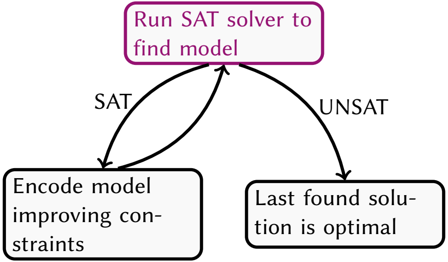
$\overline{x}_2 \vee x_3$ $\quad$ $x_2 \vee x_4 \vee r_3$

$\overline{x}_3 \vee x_4$

Objective: $min \sum_i r_i$

VeriPB proof:

derived	justification



CERTIFIED LSU SEARCH (EXAMPLE)

$\overline{x}_1 \lor x_2$ $\quad$ $\overline{x}_1 \lor \overline{x}_2 \lor r_1$

$x_1 \lor \overline{x}_2$ $\quad$ $x_1 \lor x_2 \lor r_2$

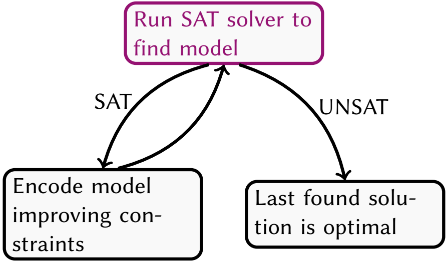
$\overline{x}_2 \lor x_3$ $\quad$ $x_2 \lor x_4 \lor r_3$

$\overline{x}_3 \lor x_4$ $\quad$ $x_2 \lor r_2$

Objective: $min \sum_i r_i$

VeriPB proof:

derived	justification



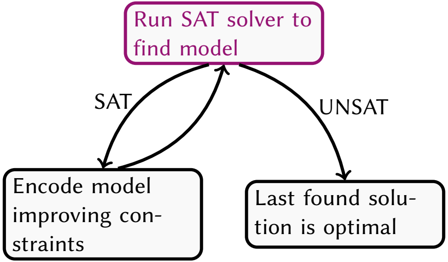
CERTIFIED LSU SEARCH (EXAMPLE)

Objective: $min \sum_i r_i$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation

$\overline{x}_1 \vee x_2$	$\overline{x}_1 \vee \overline{x}_2 \vee r_1$
$x_1 \vee \overline{x}_2$	$x_1 \vee x_2 \vee r_2$
$\overline{x}_2 \vee x_3$	$x_2 \vee x_4 \vee r_3$
$\overline{x}_3 \vee x_4$	$x_2 \vee r_2$



CERTIFIED LSU SEARCH (EXAMPLE)

$\overline{x}_1 \vee x_2$ $\overline{x}_1 \vee \overline{x}_2 \vee r_1$

$x_1 \vee \overline{x}_2$ $x_1 \vee x_2 \vee r_2$

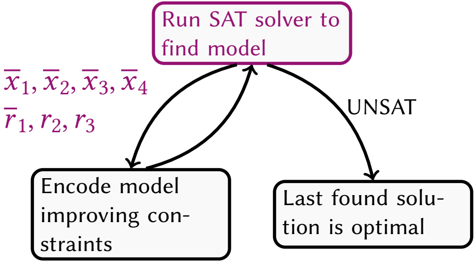
$\overline{x}_2 \vee x_3$ $x_2 \vee x_4 \vee r_3$

$\overline{x}_3 \vee x_4$ $x_2 \vee r_2$

Objective: $min \sum_i r_i$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation



CERTIFIED LSU SEARCH (EXAMPLE)

$\overline{x}_1 \vee x_2$ $\overline{x}_1 \vee \overline{x}_2 \vee r_1$

$x_1 \vee \overline{x}_2$ $x_1 \vee x_2 \vee r_2$

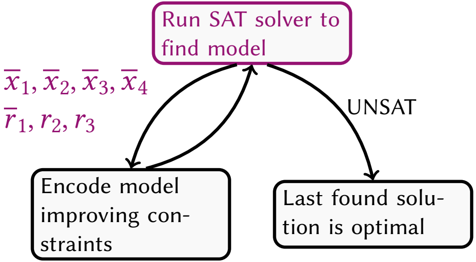
$\overline{x}_2 \vee x_3$ $x_2 \vee x_4 \vee r_3$

$\overline{x}_3 \vee x_4$ $x_2 \vee r_2$

Objective: $min \sum_i r_i$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
$\{\overline{x}_1, \ldots, \overline{x}_4, \overline{r}_1, r_2, r_3\}$	Incumbent solution



CERTIFIED LSU SEARCH (EXAMPLE)

$\overline{x}_1 \vee x_2$ $\quad$ $\overline{x}_1 \vee \overline{x}_2 \vee r_1$

$x_1 \vee \overline{x}_2$ $\quad$ $x_1 \vee x_2 \vee r_2$

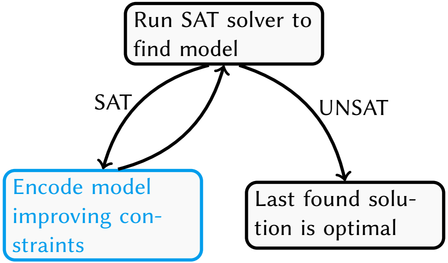
$\overline{x}_2 \vee x_3$ $\quad$ $x_2 \vee x_4 \vee r_3$

$\overline{x}_3 \vee x_4$ $\quad$ $x_2 \vee r_2$

Objective: $min \sum_i r_i$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
$\{\overline{x}_1, \ldots, \overline{x}_4, \overline{r}_1, r_2, r_3\}$	Incumbent solution
$\sum_i r_i \leq 1$	Objective Improvement Rule



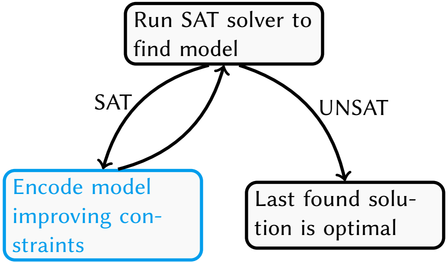
CERTIFIED LSU SEARCH (EXAMPLE)

$\overline{x}_1 \vee x_2$	$\overline{x}_1 \vee \overline{x}_2 \vee r_1$
$x_1 \vee \overline{x}_2$	$x_1 \vee x_2 \vee r_2$
$\overline{x}_2 \vee x_3$	$x_2 \vee x_4 \vee r_3$
$\overline{x}_3 \vee x_4$	$x_2 \vee r_2$

Objective: $min \sum_i r_i$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
$\{\overline{x}_1, \ldots, \overline{x}_4, \overline{r}_1, r_2, r_3\}$	Incumbent solution
$\sum_i r_i \leq 1$	Objective Improvement Rule
$\mathrm{PB}(p_1 \Leftrightarrow (\sum_i r_i \geq 1))$	Fresh variable
$\mathrm{PB}(p_2 \Leftrightarrow (\sum_i r_i \geq 2))$	



CERTIFIED LSU SEARCH (EXAMPLE)

$\overline{x}_1 \vee x_2$ $\quad$ $\overline{x}_1 \vee \overline{x}_2 \vee r_1$

$x_1 \vee \overline{x}_2$ $\quad$ $x_1 \vee x_2 \vee r_2$

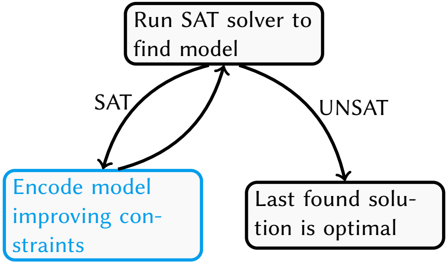
$\overline{x}_2 \vee x_3$ $\quad$ $x_2 \vee x_4 \vee r_3$

$\overline{x}_3 \vee x_4$ $\quad$ $x_2 \vee r_2$

Objective: $min \sum_i r_i$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
$\{\overline{x}_1, \ldots, \overline{x}_4, \overline{r}_1, r_2, r_3\}$	Incumbent solution
$\sum_i r_i \leq 1$	Objective Improvement Rule
$j \cdot \overline{p}_j + \sum_i r_i \geq j$	Fresh variable
$(4 - j) \cdot p_j + \sum_i \overline{r}_i \geq 4 - j$	



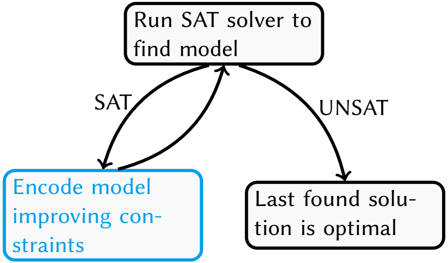
CERTIFIED LSU SEARCH (EXAMPLE)

$\overline{x}_1 \lor x_2$	$\overline{x}_1 \lor \overline{x}_2 \lor r_1$
$x_1 \lor \overline{x}_2$	$x_1 \lor x_2 \lor r_2$
$\overline{x}_2 \lor x_3$	$x_2 \lor x_4 \lor r_3$
$\overline{x}_3 \lor x_4$	$x_2 \lor r_2$

Objective: $min \sum_i r_i$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
$\{\overline{x}_1, \ldots, \overline{x}_4, \overline{r}_1, r_2, r_3\}$	Incumbent solution
$\sum_i r_i \leq 1$	Objective Improvement Rule
$j \cdot \overline{p}_j + \sum_i r_i \geq j$	Fresh variable
$(4 - j) \cdot p_j + \sum_i \overline{r}_i \geq 4 - j$	
$\text{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$	Explicit CP derivation



CERTIFIED LSU SEARCH (EXAMPLE)

Objective: $min \sum_i r_i$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
$\{\overline{x}_1, \ldots, \overline{x}_4, \overline{r}_1, r_2, r_3\}$	Incumbent solution
$\sum_i r_i \leq 1$	Objective Improvement Rule
$j \cdot \overline{p}_j + \sum_i r_i \geq j$	Fresh variable
$(4 - j) \cdot p_j + \sum_i \overline{r}_i \geq 4 - j$	
$\mathrm{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$	Explicit CP derivation

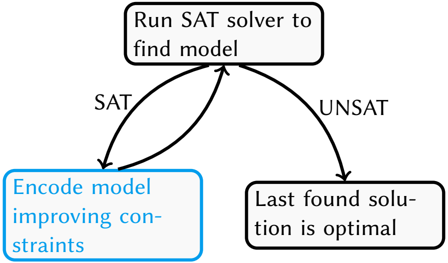
$\overline{x}_1 \lor x_2$ $\quad$ $\overline{x}_1 \lor \overline{x}_2 \lor r_1$

$x_1 \lor \overline{x}_2$ $\quad$ $x_1 \lor x_2 \lor r_2$

$\overline{x}_2 \lor x_3$ $\quad$ $x_2 \lor x_4 \lor r_3$

$\overline{x}_3 \lor x_4$ $\quad$ $x_2 \lor r_2$

$\mathrm{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$



CERTIFIED LSU SEARCH (EXAMPLE)

$\overline{x}_1 \lor x_2$ $\quad$ $\overline{x}_1 \lor \overline{x}_2 \lor r_1$

$x_1 \lor \overline{x}_2$ $\quad$ $x_1 \lor x_2 \lor r_2$

$\overline{x}_2 \lor x_3$ $\quad$ $x_2 \lor x_4 \lor r_3$

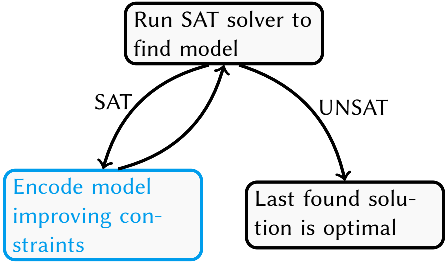
$\overline{x}_3 \lor x_4$ $\quad$ $x_2 \lor r_2$

$\mathrm{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$

Objective: $min \sum_i r_i$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
$\{\overline{x}_1, \ldots, \overline{x}_4, \overline{r}_1, r_2, r_3\}$	Incumbent solution
$\sum_i r_i \leq 1$	Objective Improvement Rule
$j \cdot \overline{p}_j + \sum_i r_i \geq j$	Fresh variable
$(4-j) \cdot p_j + \sum_i \overline{r}_i \geq 4-j$	
$\mathrm{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$	Explicit CP derivation
$\overline{p}_2 \geq 1$	Explicit CP derivation



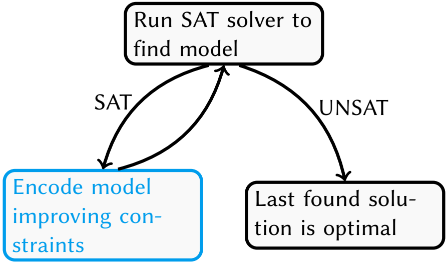
CERTIFIED LSU SEARCH (EXAMPLE)

$\overline{x}_1 \vee x_2$ $\overline{x}_1 \vee \overline{x}_2 \vee r_1$
$x_1 \vee \overline{x}_2$ $x_1 \vee x_2 \vee r_2$
$\overline{x}_2 \vee x_3$ $x_2 \vee x_4 \vee r_3$
$\overline{x}_3 \vee x_4$ $x_2 \vee r_2$
$\mathrm{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$
$\overline{p}_2$

Objective: $min \sum_i r_i$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
$\{\overline{x}_1, \ldots, \overline{x}_4, \overline{r}_1, r_2, r_3\}$	Incumbent solution
$\sum_i r_i \leq 1$	Objective Improvement Rule
$j \cdot \overline{p}_j + \sum_i r_i \geq j$	Fresh variable
$(4 - j) \cdot p_j + \sum_i \overline{r}_i \geq 4 - j$	
$\mathrm{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$	Explicit CP derivation
$\overline{p}_2 \geq 1$	Explicit CP derivation



CERTIFIED LSU SEARCH (EXAMPLE)

$\overline{x}_1 \vee x_2$ $\qquad$ $\overline{x}_1 \vee \overline{x}_2 \vee r_1$

$x_1 \vee \overline{x}_2$ $\qquad$ $x_1 \vee x_2 \vee r_2$

$\overline{x}_2 \vee x_3$ $\qquad$ $x_2 \vee x_4 \vee r_3$

$\overline{x}_3 \vee x_4$ $\qquad$ $x_2 \vee r_2$

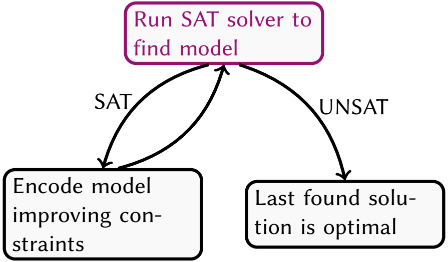
$\text{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$

$\overline{p}_2$

Objective: $min \sum_i r_i$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
$\{\overline{x}_1, \ldots, \overline{x}_4, \overline{r}_1, r_2, r_3\}$	Incumbent solution
$\sum_i r_i \leq 1$	Objective Improvement Rule
$j \cdot \overline{p}_j + \sum_i r_i \geq j$	Fresh variable
$(4-j) \cdot p_j + \sum_i \overline{r}_i \geq 4 - j$	
$\text{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$	Explicit CP derivation
$\overline{p}_2 \geq 1$	Explicit CP derivation



CERTIFIED LSU SEARCH (EXAMPLE)

$\overline{x}_1 \lor x_2$	$\overline{x}_1 \lor \overline{x}_2 \lor r_1$
$x_1 \lor \overline{x}_2$	$x_1 \lor x_2 \lor r_2$
$\overline{x}_2 \lor x_3$	$x_2 \lor x_4 \lor r_3$
$\overline{x}_3 \lor x_4$	$x_2 \lor r_2$

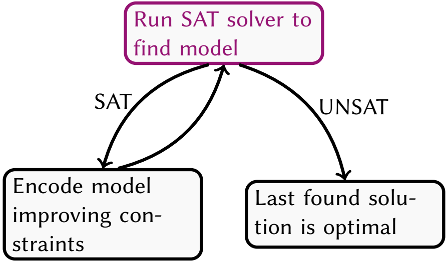
$\mathrm{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$

$\overline{p}_2$ $\quad$ x_4

Objective: $min \sum_i r_i$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
$\{\overline{x}_1, \ldots, \overline{x}_4, \overline{r}_1, r_2, r_3\}$	Incumbent solution
$\sum_i r_i \leq 1$	Objective Improvement Rule
$j \cdot \overline{p}_j + \sum_i r_i \geq j$	Fresh variable
$(4 - j) \cdot p_j + \sum_i \overline{r}_i \geq 4 - j$	
$\mathrm{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$	Explicit CP derivation
$\overline{p}_2 \geq 1$	Explicit CP derivation
$x_4 \geq 1$	Reverse Unit Propagation



CERTIFIED LSU SEARCH (EXAMPLE)

$\overline{x}_1 \lor x_2$ $\quad$ $\overline{x}_1 \lor \overline{x}_2 \lor r_1$

$x_1 \lor \overline{x}_2$ $\quad$ $x_1 \lor x_2 \lor r_2$

$\overline{x}_2 \lor x_3$ $\quad$ $x_2 \lor x_4 \lor r_3$

$\overline{x}_3 \lor x_4$ $\quad$ $x_2 \lor r_2$

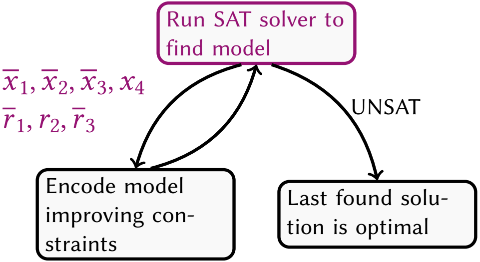
$\mathrm{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$

$\overline{p}_2$ $\quad$ x_4

Objective: $min \sum_i r_i$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
$\{\overline{x}_1, \ldots, \overline{x}_4, \overline{r}_1, r_2, r_3\}$	Incumbent solution
$\sum_i r_i \leq 1$	Objective Improvement Rule
$j \cdot \overline{p}_j + \sum_i r_i \geq j$	Fresh variable
$(4 - j) \cdot p_j + \sum_i \overline{r}_i \geq 4 - j$	
$\mathrm{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$	Explicit CP derivation
$\overline{p}_2 \geq 1$	Explicit CP derivation
$x_4 \geq 1$	Reverse Unit Propagation
$\{\overline{x}_1, \overline{x}_2, \overline{x}_3, x_4, \overline{r}_1, r_2, \overline{r}_3\}$	Incumbent solution



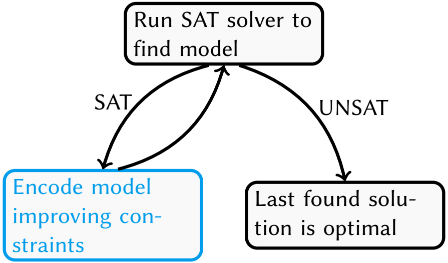
CERTIFIED LSU SEARCH (EXAMPLE)

$\overline{x}_1 \lor x_2$	$\overline{x}_1 \lor \overline{x}_2 \lor r_1$
$x_1 \lor \overline{x}_2$	$x_1 \lor x_2 \lor r_2$
$\overline{x}_2 \lor x_3$	$x_2 \lor x_4 \lor r_3$
$\overline{x}_3 \lor x_4$	$x_2 \lor r_2$
$\mathrm{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$	
$\overline{p}_2$	x_4

Objective: $min \sum_i r_i$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
$\{\overline{x}_1, \ldots, \overline{x}_4, \overline{r}_1, r_2, r_3\}$	Incumbent solution
$\sum_i r_i \leq 1$	Objective Improvement Rule
$j \cdot \overline{p}_j + \sum_i r_i \geq j$	Fresh variable
$(4-j) \cdot p_j + \sum_i \overline{r}_i \geq 4-j$	
$\mathrm{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$	Explicit CP derivation
$\overline{p}_2 \geq 1$	Explicit CP derivation
$x_4 \geq 1$	Reverse Unit Propagation
$\{\overline{x}_1, \overline{x}_2, \overline{x}_3, x_4, \overline{r}_1, r_2, \overline{r}_3\}$	Incumbent solution
$\sum_i r_i \leq 0$	Objective Improvement Rule



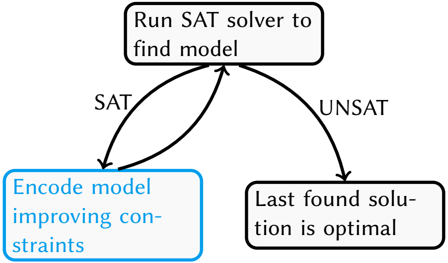
CERTIFIED LSU SEARCH (EXAMPLE)

Objective: $min \sum_i r_i$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
$\{\overline{x}_1, \ldots, \overline{x}_4, \overline{r}_1, r_2, r_3\}$	Incumbent solution
$\sum_i r_i \leq 1$	Objective Improvement Rule
$j \cdot \overline{p}_j + \sum_i r_i \geq j$	Fresh variable
$(4 - j) \cdot p_j + \sum_i \overline{r}_i \geq 4 - j$	
$\mathrm{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$	Explicit CP derivation
$\overline{p}_2 \geq 1$	Explicit CP derivation
$x_4 \geq 1$	Reverse Unit Propagation
$\{\overline{x}_1, \overline{x}_2, \overline{x}_3, x_4, \overline{r}_1, r_2, \overline{r}_3\}$	Incumbent solution
$\sum_i r_i \leq 0$	Objective Improvement Rule
$\overline{p}_1 \geq 1$	Explicit CP derivation

$\overline{x}_1 \vee x_2$	$\overline{x}_1 \vee \overline{x}_2 \vee r_1$
$x_1 \vee \overline{x}_2$	$x_1 \vee x_2 \vee r_2$
$\overline{x}_2 \vee x_3$	$x_2 \vee x_4 \vee r_3$
$\overline{x}_3 \vee x_4$	$x_2 \vee r_2$
$\mathrm{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$	
$\overline{p}_2$	x_4
$\overline{p}_1$	



CERTIFIED LSU SEARCH (EXAMPLE)

$\overline{x}_1 \vee x_2$ $\overline{x}_1 \vee \overline{x}_2 \vee r_1$

$x_1 \vee \overline{x}_2$ $x_1 \vee x_2 \vee r_2$

$\overline{x}_2 \vee x_3$ $x_2 \vee x_4 \vee r_3$

$\overline{x}_3 \vee x_4$ $x_2 \vee r_2$

$\mathrm{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$

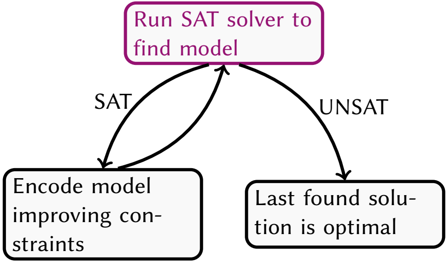
$\overline{p}_2$ x_4

$\overline{p}_1$

Objective: $min \sum_i r_i$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
$\{\overline{x}_1, \ldots, \overline{x}_4, \overline{r}_1, r_2, r_3\}$	Incumbent solution
$\sum_i r_i \leq 1$	Objective Improvement Rule
$j \cdot \overline{p}_j + \sum_i r_i \geq j$	Fresh variable
$(4-j) \cdot p_j + \sum_i \overline{r}_i \geq 4-j$	
$\mathrm{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$	Explicit CP derivation
$\overline{p}_2 \geq 1$	Explicit CP derivation
$x_4 \geq 1$	Reverse Unit Propagation
$\{\overline{x}_1, \overline{x}_2, \overline{x}_3, x_4, \overline{r}_1, r_2, \overline{r}_3\}$	Incumbent solution
$\sum_i r_i \leq 0$	Objective Improvement Rule
$\overline{p}_1 \geq 1$	Explicit CP derivation



CERTIFIED LSU SEARCH (EXAMPLE)

Objective: *min* $\sum_i r_i$

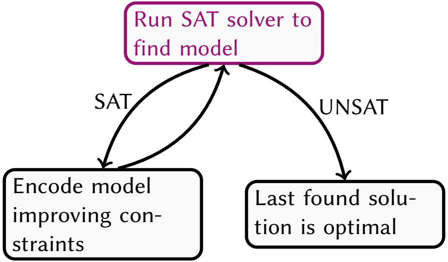
VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
$\{\overline{x}_1, \ldots, \overline{x}_4, \overline{r}_1, r_2, r_3\}$	Incumbent solution
$\sum_i r_i \leq 1$	Objective Improvement Rule
$j \cdot \overline{p}_j + \sum_i r_i \geq j$	Fresh variable
$(4 - j) \cdot p_j + \sum_i \overline{r}_i \geq 4 - j$	
$\mathrm{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$	Explicit CP derivation
$\overline{p}_2 \geq 1$	Explicit CP derivation
$x_4 \geq 1$	Reverse Unit Propagation
$\{\overline{x}_1, \overline{x}_2, \overline{x}_3, x_4, \overline{r}_1, r_2, \overline{r}_3\}$	Incumbent solution
$\sum_i r_i \leq 0$	Objective Improvement Rule
$\overline{p}_1 \geq 1$	Explicit CP derivation
$0 \geq 1$	Reverse Unit Propagation

$\overline{x}_1 \vee x_2$	$\overline{x}_1 \vee \overline{x}_2 \vee r_1$
$x_1 \vee \overline{x}_2$	$x_1 \vee x_2 \vee r_2$
$\overline{x}_2 \vee x_3$	$x_2 \vee x_4 \vee r_3$
$\overline{x}_3 \vee x_4$	$x_2 \vee r_2$

$\mathrm{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$

$\overline{p}_2$	x_4
$\overline{p}_1$	$\perp$



CERTIFIED LSU SEARCH (EXAMPLE)

Objective: $min \sum_i r_i$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
$\{\overline{x}_1, \ldots, \overline{x}_4, \overline{r}_1, r_2, r_3\}$	Incumbent solution
$\sum_i r_i \leq 1$	Objective Improvement Rule
$j \cdot \overline{p}_j + \sum_i r_i \geq j$	Fresh variable
$(4-j) \cdot p_j + \sum_i \overline{r}_i \geq 4-j$	
$\mathrm{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$	Explicit CP derivation
$\overline{p}_2 \geq 1$	Explicit CP derivation
$x_4 \geq 1$	Reverse Unit Propagation
$\{\overline{x}_1, \overline{x}_2, \overline{x}_3, x_4, \overline{r}_1, r_2, \overline{r}_3\}$	Incumbent solution
$\sum_i r_i \leq 0$	Objective Improvement Rule
$\overline{p}_1 \geq 1$	Explicit CP derivation
$0 \geq 1$	Reverse Unit Propagation

$\overline{x}_1 \vee x_2$

$x_1 \vee \overline{x}_2$

$\overline{x}_2 \vee x_3$

$\overline{x}_3 \vee x_4$

$\mathrm{CNF}(p_j \Leftrightarrow (\sum_i r_i \geq j))$

$\overline{p}_2$

$\overline{p}_1$

$\overline{x}_1 \vee \overline{x}_2 \vee r_1$

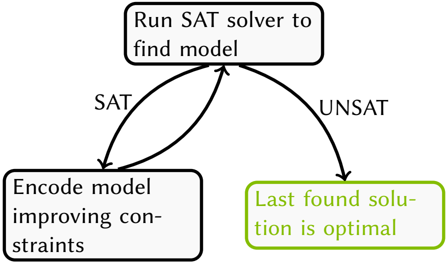
$x_1 \vee x_2 \vee r_2$

$x_2 \vee x_4 \vee r_3$

$x_2 \vee r_2$

x_4

$\perp$



LSU EXAMPLE IN VERIPB SYNTAX

```
pseudo-Boolean proof version 2.0
f 7
* Clauses derived by solver
rup 1 x1 1 r2 >= 1 ;
* Log incumbent solution
soli ~x1 ~x2 ~x3 ~x4 ~r1 r2 r3
* introduce fresh variables
red 2 ~p2 1 r1 1 r2 1 r3  >= 2 ; p2 -> 0 ;
red 2 p2 1 ~r1 1 ~r2 1 ~r3  >= 2; p2 -> 1 ;
red 1 ~p1 1 r1 1 r2 1 r3  >= 1; p1 -> 0 ;
red 3 p1 1 ~r1 1 ~r2 1 ~r3 >= 3; p1 -> 1 ;
* Derive CNF encoding of totalizer
. . .  - coming soon
* Derive counter falsity
pol 9 10 + s
* Clauses derived by solver
rup 1 x4 >= 1 ;
* Log incumbent solution
soli ~x1 ~x2 ~x3 x4 ~r1 r2 ~r3
* Derive counter falsity
pol -1 12 +
* Inconsistency derived by solver
rup >= 1 ;
* Conclusion
output NONE
conclusion BOUNDS 1 1
end pseudo-Boolean proof
```

CERTIFIED ENCODING OF THE MODEL-IMPROVING CONSTRAINT

How to encode $p_j \Leftrightarrow \sum_i r_i \geq j$ in CNF?

CERTIFIED ENCODING OF THE MODEL-IMPROVING CONSTRAINT

How to encode $p_j \Leftrightarrow \sum_i r_i \geq j$ in CNF?

Different MaxSAT solvers use different PB-to-CNF encodings, e.g.,

- Totalizer Encoding [BB03]
- Binary Adder [War98]
- Modulo-Based Totalizer [OLH+13]
- Sorting Networks [ES06, ANOR09]
- (Dynamic) Polynomial Watchdog (DPW) [PRB18]

CERTIFIED ENCODING OF THE MODEL-IMPROVING CONSTRAINT

How to encode $p_j \Leftrightarrow \sum_i r_i \geq j$ in CNF?

Different MaxSAT solvers use different PB-to-CNF encodings, e.g.,

- Totalizer Encoding [BB03]
- Binary Adder [War98]
- Modulo-Based Totalizer [OLH+13]
- Sorting Networks [ES06, ANOR09]
- (Dynamic) Polynomial Watchdog (DPW) [PRB18]

Totalizer encoding demonstrated here; ideas generalize to other encodings [Van23]

CERTIFIED ENCODING OF THE MODEL-IMPROVING CONSTRAINT

How to encode $p_j \Leftrightarrow \sum_i r_i \geq j$ in CNF?

Different MaxSAT solvers use different PB-to-CNF encodings, e.g.,

- Totalizer Encoding [BB03]
- Binary Adder [War98]
- Modulo-Based Totalizer [OLH+13]
- Sorting Networks [ES06, ANOR09]
- (Dynamic) Polynomial Watchdog (DPW) [PRB18]

Totalizer encoding demonstrated here; ideas generalize to other encodings [Van23]

Except... DPW turns out to use complicated without-loss-of-generality reasoning [BBN+24]

TOTALIZER ENCODING OF CARDINALITY CONSTRAINTS

How to encode $p^I_j \Leftrightarrow \sum_{i \in I} r_i \geq j$?

- Totalizer encoding [BB03]

TOTALIZER ENCODING OF CARDINALITY CONSTRAINTS

How to encode $p^I_j \Leftrightarrow \sum_{i \in I} r_i \geq j$?

- Totalizer encoding [BB03]
- Create binary tree (leaves are the r_i); and introduce counter variables in all nodes

TOTALIZER ENCODING OF CARDINALITY CONSTRAINTS

How to encode $p^I_j \Leftrightarrow \sum_{i \in I} r_i \geq j$?

- Totalizer encoding [BB03]
- Create binary tree (leaves are the r_i); and introduce counter variables in all nodes
- Example: $I = \{1, \cdots, 8\}$, $I_1 = \{1, \cdots, 4\}$ and $I_2 = \{5, \cdots, 8\}$

$$p^I_1, p^I_2, p^I_3, p^I_4, p^I_5, p^I_6, p^I_7, p^I_8$$

$$p^{I_1}_1, p^{I_1}_2, p^{I_1}_3, p^{I_1}_4 \qquad p^{I_2}_1, p^{I_2}_2, p^{I_2}_3, p^{I_2}_4$$

TOTALIZER ENCODING OF CARDINALITY CONSTRAINTS

How to encode $p^I_j \Leftrightarrow \sum_{i \in I} r_i \geq j$?

- Totalizer encoding [BB03]
- Create binary tree (leaves are the r_i); and introduce counter variables in all nodes
- Example: $I = \{1, \cdots, 8\}$, $I_1 = \{1, \cdots, 4\}$ and $I_2 = \{5, \cdots, 8\}$

$$p^I_1, p^I_2, p^I_3, p^I_4, p^I_5, p^I_6, p^I_7, p^I_8$$

$$p^{I_1}_1, p^{I_1}_2, p^{I_1}_3, p^{I_1}_4 \qquad p^{I_2}_1, p^{I_2}_2, p^{I_2}_3, p^{I_2}_4$$

Clauses encoding $p^I_6 \Leftarrow \sum_{i \in I} r_i \geq 6$:

$$\left(p^{I_1}_2 \wedge p^{I_2}_4\right) \Rightarrow p^I_6 \qquad \left(p^{I_1}_3 \wedge p^{I_2}_3\right) \Rightarrow p^I_6 \qquad \left(p^{I_1}_4 \wedge p^{I_2}_2\right) \Rightarrow p^I_6$$

TOTALIZER ENCODING OF CARDINALITY CONSTRAINTS

How to encode $p_j^I \Leftrightarrow \sum_{i \in I} r_i \geq j$?

- Totalizer encoding [BB03]
- Create binary tree (leaves are the r_i); and introduce counter variables in all nodes
- Example: $I = \{1, \cdots, 8\}$, $I_1 = \{1, \cdots, 4\}$ and $I_2 = \{5, \cdots, 8\}$

$$p_1^I, p_2^I, p_3^I, p_4^I, p_5^I, p_6^I, p_7^I, p_8^I$$

$$p_1^{I_1}, p_2^{I_1}, p_3^{I_1}, p_4^{I_1} \qquad\qquad p_1^{I_2}, p_2^{I_2}, p_3^{I_2}, p_4^{I_2}$$

Clauses encoding $p_6^I \Leftarrow \sum_{i \in I} r_i \geq 6$:

$$\overline{p}_2^{I_1} \vee \overline{p}_4^{I_2} \vee p_6^I \qquad \overline{p}_3^{I_1} \vee \overline{p}_3^{I_2} \vee p_6^I \qquad \overline{p}_4^{I_1} \vee \overline{p}_2^{I_2} \vee p_6^I$$

TOTALIZER ENCODING OF CARDINALITY CONSTRAINTS

How to encode $p_j^I \Leftrightarrow \sum_{i \in I} r_i \geq j$?

- Totalizer encoding [BB03]
- Create binary tree (leaves are the r_i); and introduce counter variables in all nodes
- Example: $I = \{1, \cdots, 8\}$, $I_1 = \{1, \cdots, 4\}$ and $I_2 = \{5, \cdots, 8\}$

$$p_1^I, p_2^I, p_3^I, p_4^I, p_5^I, p_6^I, p_7^I, p_8^I$$

$$p_1^{I_1}, p_2^{I_1}, p_3^{I_1}, p_4^{I_1} \qquad\qquad p_1^{I_2}, p_2^{I_2}, p_3^{I_2}, p_4^{I_2}$$

Clauses encoding $p_6^I \Leftarrow \sum_{i \in I} r_i \geq 6$:

$$\overline{p}_2^{I_1} \lor \overline{p}_4^{I_2} \lor p_6^I \qquad \overline{p}_3^{I_1} \lor \overline{p}_3^{I_2} \lor p_6^I \qquad \overline{p}_4^{I_1} \lor \overline{p}_2^{I_2} \lor p_6^I$$

Clauses encoding $p_6^I \Rightarrow \sum_{i \in I} r_i \geq 6$:

$$\overline{p}_2^{I_1} \Rightarrow \overline{p}_6^I \qquad \left(\overline{p}_3^{I_1} \land \overline{p}_4^{I_2}\right) \Rightarrow \overline{p}_6^I \qquad \left(\overline{p}_4^{I_1} \land \overline{p}_3^{I_2}\right) \Rightarrow \overline{p}_6^I \qquad \overline{p}_2^{I_2} \Rightarrow \overline{p}_6^I$$

TOTALIZER ENCODING OF CARDINALITY CONSTRAINTS

How to encode $p_j^I \Leftrightarrow \sum_{i \in I} r_i \geq j$?

- Totalizer encoding [BB03]
- Create binary tree (leaves are the r_i); and introduce counter variables in all nodes
- Example: $I = \{1, \cdots, 8\}$, $I_1 = \{1, \cdots, 4\}$ and $I_2 = \{5, \cdots, 8\}$

$$p_1^I, p_2^I, p_3^I, p_4^I, p_5^I, p_6^I, p_7^I, p_8^I$$

$$p_1^{I_1}, p_2^{I_1}, p_3^{I_1}, p_4^{I_1} \qquad\qquad p_1^{I_2}, p_2^{I_2}, p_3^{I_2}, p_4^{I_2}$$

Clauses encoding $p_6^I \Leftarrow \sum_{i \in I} r_i \geq 6$:

$$\overline{p}_2^{I_1} \vee \overline{p}_4^{I_2} \vee p_6^I \qquad \overline{p}_3^{I_1} \vee \overline{p}_3^{I_2} \vee p_6^I \qquad \overline{p}_4^{I_1} \vee \overline{p}_2^{I_2} \vee p_6^I$$

Clauses encoding $p_6^I \Rightarrow \sum_{i \in I} r_i \geq 6$:

$$p_2^{I_1} \vee \overline{p}_6^I \qquad p_3^{I_1} \vee p_4^{I_2} \vee \overline{p}_6^I \qquad p_4^{I_1} \vee p_3^{I_2} \vee \overline{p}_6^I \qquad p_2^{I_2} \vee \overline{p}_6^I$$

CERTIFYING THE TOTALIZER ENCODING USING CUTTING PLANES

- To be derived: $\overline{p}_4^{I_1} \vee \overline{p}_2^{I_2} \vee p_6^{I}$

CERTIFYING THE TOTALIZER ENCODING USING CUTTING PLANES

- To be derived: $\overline{p}_4^{I_1} \vee \overline{p}_2^{I_2} \vee p_6^{I}$
- Counting variables introduced using

$$4 \cdot \overline{p}_4^{I_1} + \sum_{i \in I_1} r_i \geq 4$$

$$2 \cdot \overline{p}_2^{I_2} + \sum_{i \in I_2} r_i \geq 2$$

$$3 \cdot p_6^{I} + \sum_{i \in I} \overline{r}_i \geq 3$$

CERTIFYING THE TOTALIZER ENCODING USING CUTTING PLANES

- To be derived: $\overline{p}_4^{I_1} \vee \overline{p}_2^{I_2} \vee p_6^{I}$
- Counting variables introduced using

$$4 \cdot \overline{p}_4^{I_1} + \sum_{i \in I_1} r_i \geq 4$$

$$2 \cdot \overline{p}_2^{I_2} + \sum_{i \in I_2} r_i \geq 2$$

$$3 \cdot p_6^{I} + \sum_{i \in I} \overline{r}_i \geq 3$$

- Adding these three constraints yields

$$4 \cdot \overline{p}_4^{I_1} + 2 \cdot \overline{p}_2^{I_2} + 3 \cdot p_6^{I} + 8 \geq 9$$

CERTIFYING THE TOTALIZER ENCODING USING CUTTING PLANES

- To be derived: $\overline{p}_4^{I_1} \vee \overline{p}_2^{I_2} \vee p_6^{I}$
- Counting variables introduced using

$$4 \cdot \overline{p}_4^{I_1} + \sum_{i \in I_1} r_i \geq 4$$

$$2 \cdot \overline{p}_2^{I_2} + \sum_{i \in I_2} r_i \geq 2$$

$$3 \cdot p_6^{I} + \sum_{i \in I} \overline{r}_i \geq 3$$

- Adding these three constraints yields

$$4 \cdot \overline{p}_4^{I_1} + 2 \cdot \overline{p}_2^{I_2} + 3 \cdot p_6^{I} + \cancel{8} \geq \cancel{9}\ 1$$

CERTIFYING THE TOTALIZER ENCODING USING CUTTING PLANES

- To be derived: $\overline{p}_4^{I_1} \vee \overline{p}_2^{I_2} \vee p_6^{I}$
- Counting variables introduced using

$$4 \cdot \overline{p}_4^{I_1} + \sum_{i \in I_1} r_i \geq 4$$

$$2 \cdot \overline{p}_2^{I_2} + \sum_{i \in I_2} r_i \geq 2$$

$$3 \cdot p_6^{I} + \sum_{i \in I} \overline{r}_i \geq 3$$

- Adding these three constraints and saturating yields

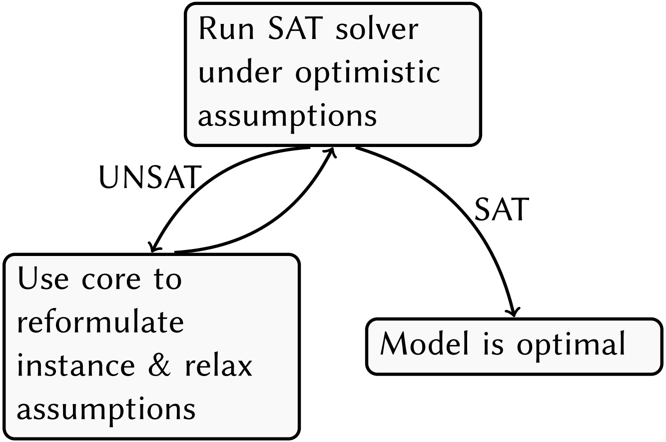
$$\cancel{4} \cdot \overline{p}_4^{I_1} + \cancel{2} \cdot \overline{p}_2^{I_2} + \cancel{3} \cdot p_6^{I} + \cancel{8} \geq \cancel{9}\ 1$$

COMPLETE LSU EXAMPLE IN VERIPB SYNTAX

```
pseudo-Boolean proof version 2.0
f 7
* Clauses derived by solver
rup 1 x1 1 r2 >= 1 ;
* Log incumbent solution
soli ~x1 ~x2 ~x3 ~x4 ~r1 r2 r3
* introduce fresh variables
red 2 ~p2 1 r1 1 r2 1 r3  >= 2 ; p2 -> 0 ;
red 2 p2 1 ~r1 1 ~r2 1 ~r3  >= 2; p2 -> 1 ;
red 1 ~p1 1 r1 1 r2 1 r3  >= 1; p1 -> 0 ;
red 3 p1 1 ~r1 1 ~r2 1 ~r3 >= 3; p1 -> 1 ;
* Auxiliary variables for CNF encoding
red 2 ~p_1-2_2 1 r1 1 r2  >= 2 ; p_1-2_2 -> 0 ;
red 1 p_1-2_2 1 ~r1 1 ~r2 >= 1; p_1-2_2 -> 1 ;
red 1 ~p_1-2_1 1 r1 1 r2  >= 1; p_1-2_1 -> 0 ;
red 2 p_1-2_1 1 ~r1 1 ~r2 >= 2; p_1-2_1 -> 1 ;
* Cutting planes derivation of totalizer clauses
pol 10 15 + s
pol 10 17 + ~r3 + s
pol 11 14 + r3 + s
pol 11 16 + s
pol 12 17 + s
pol 13 16 + r3 + s
pol 13 r1 + r2 + s
* Derive counter falsity
pol 9 10 + s
* Clauses derived by solver
rup 1 x4 >= 1 ;
* Log incumbent solution
soli ~x1 ~x2 ~x3 x4 ~r1 r2 ~r3
* Derive counter falsity
pol -1 12 +
* Inconsistency derived by solver
rup >= 1 ;
* Conclusion
output NONE
conclusion BOUNDS 1 1
end pseudo-Boolean proof
```

OUTLINE

CORE-GUIDED SEARCH



CERTIFIED CORE-GUIDED SEARCH (EXAMPLE)

Objective (*min*): $r_1 + r_2 + r_3$

$\overline{x}_1 \vee x_2$

$x_1 \vee \overline{x}_2$

$\overline{x}_2 \vee x_3$

$\overline{x}_3 \vee x_4$

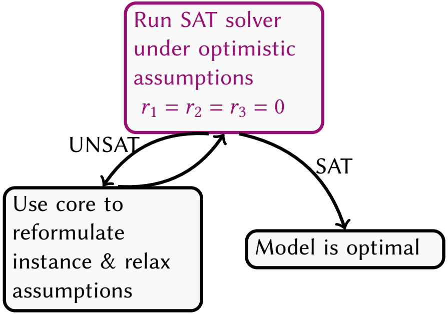
$\overline{x}_1 \vee \overline{x}_2 \vee r_1$

$x_1 \vee x_2 \vee r_2$

$x_2 \vee x_4 \vee r_3$

VeriPB proof:

derived	justification



CERTIFIED CORE-GUIDED SEARCH (EXAMPLE)

Objective (*min*): $r_1 + r_2 + r_3$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
Core returned by solver:	
$r_1 + r_2 \geq 1$	Reverse Unit Propagation

$\overline{x}_1 \vee x_2$

$x_1 \vee \overline{x}_2$

$\overline{x}_2 \vee x_3$

$\overline{x}_3 \vee x_4$

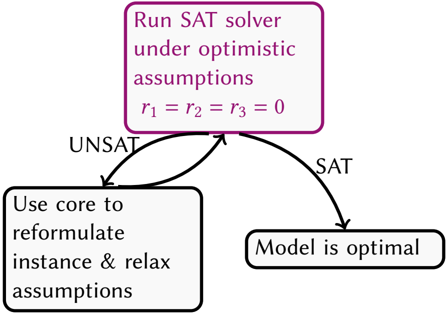
$r_1 \vee r_2$

$\overline{x}_1 \vee \overline{x}_2 \vee r_1$

$x_1 \vee x_2 \vee r_2$

$x_2 \vee x_4 \vee r_3$

$x_2 \vee r_2$



CERTIFIED CORE-GUIDED SEARCH (EXAMPLE)

Objective (*min*): $r_1 + r_2 + r_3$

$\overline{x}_1 \vee x_2$

$x_1 \vee \overline{x}_2$

$\overline{x}_2 \vee x_3$

$\overline{x}_3 \vee x_4$

$r_1 \vee r_2$

$\overline{x}_1 \vee \overline{x}_2 \vee r_1$

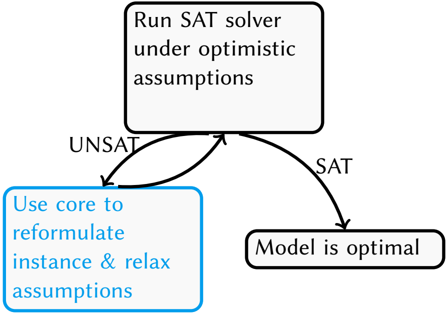
$x_1 \vee x_2 \vee r_2$

$x_2 \vee x_4 \vee r_3$

$x_2 \vee r_2$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
Core returned by solver:	
$r_1 + r_2 \geq 1$	Reverse Unit Propagation
$p_2 \Leftrightarrow (r_1 + r_2 \geq 2)$	Fresh variable



CERTIFIED CORE-GUIDED SEARCH (EXAMPLE)

Objective (*min*): $r_1 + r_2 + r_3$

$\overline{x}_1 \lor x_2$ $\quad$ $\overline{x}_1 \lor \overline{x}_2 \lor r_1$

$x_1 \lor \overline{x}_2$ $\quad$ $x_1 \lor x_2 \lor r_2$

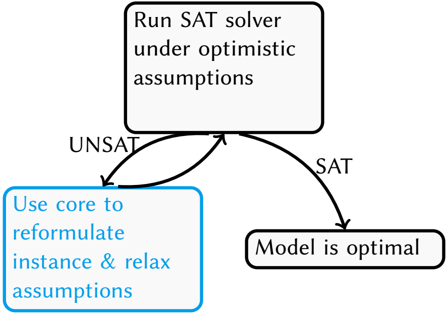
$\overline{x}_2 \lor x_3$ $\quad$ $x_2 \lor x_4 \lor r_3$

$\overline{x}_3 \lor x_4$ $\quad$ $x_2 \lor r_2$

$r_1 \lor r_2$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
Core returned by solver:	
$r_1 + r_2 \geq 1$	Reverse Unit Propagation
$2 \cdot \overline{p}_2 + r_1 + r_2 \geq 2$	Fresh variable
$p_2 + \overline{r}_1 + \overline{r}_2 \geq 1$	



CERTIFIED CORE-GUIDED SEARCH (EXAMPLE)

Objective (*min*): $r_1 + r_2 + r_3$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
Core returned by solver:	
$r_1 + r_2 \geq 1$	Reverse Unit Propagation
$2 \cdot \overline{p}_2 + r_1 + r_2 \geq 2$	Fresh variable
$p_2 + \overline{r}_1 + \overline{r}_2 \geq 1$	
$\mathrm{CNF}(p_2 \Leftrightarrow (r_1 + r_2 \geq 2))$	Explicit CP derivation

$\overline{x}_1 \vee x_2$ $\quad$ $\overline{x}_1 \vee \overline{x}_2 \vee r_1$

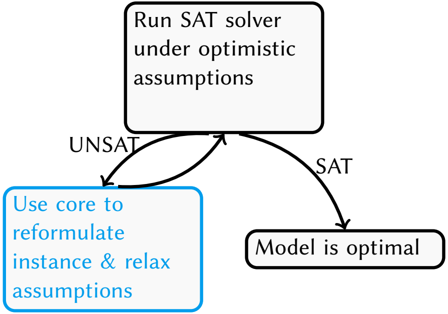
$x_1 \vee \overline{x}_2$ $\quad$ $x_1 \vee x_2 \vee r_2$

$\overline{x}_2 \vee x_3$ $\quad$ $x_2 \vee x_4 \vee r_3$

$\overline{x}_3 \vee x_4$ $\quad$ $x_2 \vee r_2$

$r_1 \vee r_2$

$\mathrm{CNF}(p_2 \Leftrightarrow (r_1 + r_2 \geq 2))$



CERTIFIED CORE-GUIDED SEARCH (EXAMPLE)

Objective (*min*): $r_1 + r_2 + r_3 = 1 + p_2 + r_3$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
Core returned by solver:	
$r_1 + r_2 \geq 1$	Reverse Unit Propagation
$2 \cdot \overline{p}_2 + r_1 + r_2 \geq 2$	Fresh variable
$p_2 + \overline{r}_1 + \overline{r}_2 \geq 1$	
$\mathrm{CNF}(p_2 \Leftrightarrow (r_1 + r_2 \geq 2))$	Explicit CP derivation
$r_1 + r_2 = 1 + p_2$	Explicit CP derivation

$\overline{x}_1 \lor x_2$ $\quad$ $\overline{x}_1 \lor \overline{x}_2 \lor r_1$

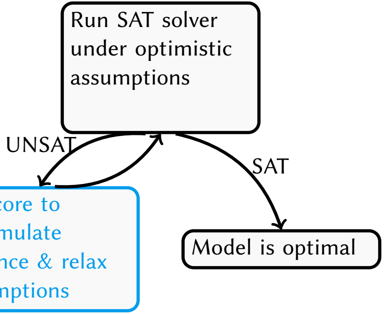
$x_1 \lor \overline{x}_2$ $\quad$ $x_1 \lor x_2 \lor r_2$

$\overline{x}_2 \lor x_3$ $\quad$ $x_2 \lor x_4 \lor r_3$

$\overline{x}_3 \lor x_4$ $\quad$ $x_2 \lor r_2$

$r_1 \lor r_2$

$\mathrm{CNF}(p_2 \Leftrightarrow (r_1 + r_2 \geq 2))$



CERTIFIED CORE-GUIDED SEARCH (EXAMPLE)

Objective (*min*): $r_1 + r_2 + r_3 = 1 + p_2 + r_3$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
Core returned by solver:	
$r_1 + r_2 \geq 1$	Reverse Unit Propagation
$2 \cdot \overline{p}_2 + r_1 + r_2 \geq 2$	Fresh variable
$p_2 + \overline{r}_1 + \overline{r}_2 \geq 1$	
$\mathrm{CNF}(p_2 \Leftrightarrow (r_1 + r_2 \geq 2))$	Explicit CP derivation
$r_1 + r_2 = 1 + p_2$	Explicit CP derivation

$\overline{x}_1 \vee x_2$ $\quad$ $\overline{x}_1 \vee \overline{x}_2 \vee r_1$

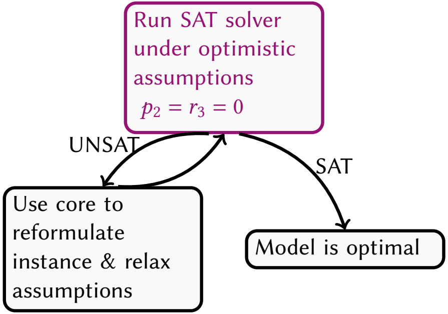
$x_1 \vee \overline{x}_2$ $\quad$ $x_1 \vee x_2 \vee r_2$

$\overline{x}_2 \vee x_3$ $\quad$ $x_2 \vee x_4 \vee r_3$

$\overline{x}_3 \vee x_4$ $\quad$ $x_2 \vee r_2$

$r_1 \vee r_2$

$\mathrm{CNF}(p_2 \Leftrightarrow (r_1 + r_2 \geq 2))$



CERTIFIED CORE-GUIDED SEARCH (EXAMPLE)

Objective (*min*): $r_1 + r_2 + r_3 = 1 + p_2 + r_3$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
Core returned by solver:	
$r_1 + r_2 \geq 1$	Reverse Unit Propagation
$2 \cdot \overline{p}_2 + r_1 + r_2 \geq 2$	Fresh variable
$p_2 + \overline{r}_1 + \overline{r}_2 \geq 1$	
$\mathrm{CNF}(p_2 \Leftrightarrow (r_1 + r_2 \geq 2))$	Explicit CP derivation
$r_1 + r_2 = 1 + p_2$	Explicit CP derivation
$\{x_1, x_2, x_3, x_4, r_1, \overline{r}_2, \overline{r}_3\}$	Solution

$\overline{x}_1 \vee x_2$ $\quad$ $\overline{x}_1 \vee \overline{x}_2 \vee r_1$

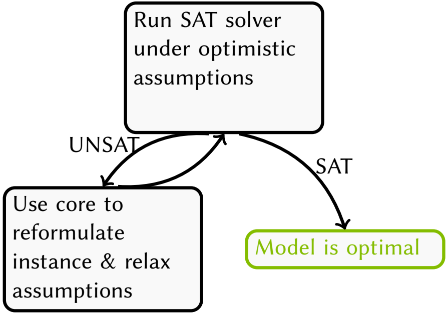
$x_1 \vee \overline{x}_2$ $\quad$ $x_1 \vee x_2 \vee r_2$

$\overline{x}_2 \vee x_3$ $\quad$ $x_2 \vee x_4 \vee r_3$

$\overline{x}_3 \vee x_4$ $\quad$ $x_2 \vee r_2$

$r_1 \vee r_2$

$\mathrm{CNF}(p_2 \Leftrightarrow (r_1 + r_2 \geq 2))$



CERTIFIED CORE-GUIDED SEARCH (EXAMPLE)

Objective (*min*): $r_1 + r_2 + r_3 = 1 + p_2 + r_3$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
Core returned by solver:	
$r_1 + r_2 \geq 1$	Reverse Unit Propagation
$2 \cdot \overline{p}_2 + r_1 + r_2 \geq 2$	Fresh variable
$p_2 + \overline{r}_1 + \overline{r}_2 \geq 1$	
$\mathrm{CNF}(p_2 \Leftrightarrow (r_1 + r_2 \geq 2))$	Explicit CP derivation
$r_1 + r_2 = 1 + p_2$	Explicit CP derivation
$\{x_1, x_2, x_3, x_4, r_1, \overline{r}_2, \overline{r}_3\}$	Solution
$\overline{r}_1 + \overline{r}_2 + \overline{r}_3 \geq 3$	Objective Improvement

$\overline{x}_1 \vee x_2$ $\quad$ $\overline{x}_1 \vee \overline{x}_2 \vee r_1$

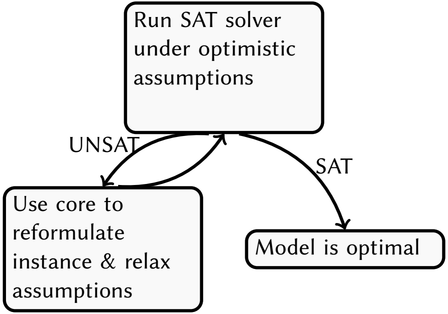
$x_1 \vee \overline{x}_2$ $\quad$ $x_1 \vee x_2 \vee r_2$

$\overline{x}_2 \vee x_3$ $\quad$ $x_2 \vee x_4 \vee r_3$

$\overline{x}_3 \vee x_4$ $\quad$ $x_2 \vee r_2$

$r_1 \vee r_2$

$\mathrm{CNF}(p_2 \Leftrightarrow (r_1 + r_2 \geq 2))$



CERTIFIED CORE-GUIDED SEARCH (EXAMPLE)

Objective (*min*): $r_1 + r_2 + r_3 = 1 + p_2 + r_3$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
Core returned by solver:	
$r_1 + r_2 \geq 1$	Reverse Unit Propagation
$2 \cdot \overline{p}_2 + r_1 + r_2 \geq 2$	Fresh variable
$p_2 + \overline{r}_1 + \overline{r}_2 \geq 1$	
$\mathrm{CNF}(p_2 \Leftrightarrow (r_1 + r_2 \geq 2))$	Explicit CP derivation
$r_1 + r_2 = 1 + p_2$	Explicit CP derivation
$\{x_1, x_2, x_3, x_4, r_1, \overline{r}_2, \overline{r}_3\}$	Solution
$\overline{r}_1 + \overline{r}_2 + \overline{r}_3 \geq 3$	Objective Improvement
$0 \geq 1$	Explicit CP derivation

$\overline{x}_1 \vee x_2$

$x_1 \vee \overline{x}_2$

$\overline{x}_2 \vee x_3$

$\overline{x}_3 \vee x_4$

$r_1 \vee r_2$

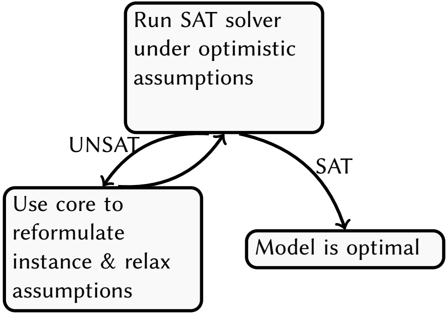
$\mathrm{CNF}(p_2 \Leftrightarrow (r_1 + r_2 \geq 2))$

$\overline{x}_1 \vee \overline{x}_2 \vee r_1$

$x_1 \vee x_2 \vee r_2$

$x_2 \vee x_4 \vee r_3$

$x_2 \vee r_2$



CERTIFIED CORE-GUIDED SEARCH (EXAMPLE)

Objective (*min*): $r_1 + r_2 + r_3 = 1 + p_2 + r_3$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
Core returned by solver:	
$r_1 + r_2 \geq 1$	Reverse Unit Propagation
$2 \cdot \overline{p}_2 + r_1 + r_2 \geq 2$	Fresh variable
$p_2 + \overline{r}_1 + \overline{r}_2 \geq 1$	
$\mathrm{CNF}(p_2 \Leftrightarrow (r_1 + r_2 \geq 2))$	Explicit CP derivation
$r_1 + r_2 = 1 + p_2$	Explicit CP derivation
$\{x_1, x_2, x_3, x_4, r_1, \overline{r}_2, \overline{r}_3\}$	Solution
$\overline{r}_1 + \overline{r}_2 + \overline{r}_3 \geq 3$	Objective Improvement
$0 \geq 1$	Explicit CP derivation

Explicit CP derivations:

CNF encoding (totalizer): cf. LSU

CERTIFIED CORE-GUIDED SEARCH (EXAMPLE)

Objective (*min*): $r_1 + r_2 + r_3 = 1 + p_2 + r_3$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
Core returned by solver:	
$r_1 + r_2 \geq 1$	Reverse Unit Propagation
$2 \cdot \overline{p}_2 + r_1 + r_2 \geq 2$	Fresh variable
$p_2 + \overline{r}_1 + \overline{r}_2 \geq 1$	
$\mathrm{CNF}(p_2 \Leftrightarrow (r_1 + r_2 \geq 2))$	Explicit CP derivation
$r_1 + r_2 = 1 + p_2$	Explicit CP derivation
$\{x_1, x_2, x_3, x_4, r_1, \overline{r}_2, \overline{r}_3\}$	Solution
$\overline{r}_1 + \overline{r}_2 + \overline{r}_3 \geq 3$	Objective Improvement
$0 \geq 1$	Explicit CP derivation

Explicit CP derivations:

CNF encoding (totalizer): cf. LSU

Adding up definition of p_2 and core constraint yields

$$2 \cdot \overline{p}_2 + 2 \cdot r_1 + 2 \cdot r_2 \geq 3 \ .$$

CERTIFIED CORE-GUIDED SEARCH (EXAMPLE)

Objective (*min*): $r_1 + r_2 + r_3 = 1 + p_2 + r_3$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
Core returned by solver:	
$r_1 + r_2 \geq 1$	Reverse Unit Propagation
$2 \cdot \overline{p}_2 + r_1 + r_2 \geq 2$	Fresh variable
$p_2 + \overline{r}_1 + \overline{r}_2 \geq 1$	
$\mathrm{CNF}(p_2 \Leftrightarrow (r_1 + r_2 \geq 2))$	Explicit CP derivation
$r_1 + r_2 = 1 + p_2$	Explicit CP derivation
$\{x_1, x_2, x_3, x_4, r_1, \overline{r}_2, \overline{r}_3\}$	Solution
$\overline{r}_1 + \overline{r}_2 + \overline{r}_3 \geq 3$	Objective Improvement
$0 \geq 1$	Explicit CP derivation

Explicit CP derivations:

CNF encoding (totalizer): cf. LSU

Adding up definition of p_2 and core constraint and dividing by 2 yields

$$\cancel{2} \cdot \overline{p}_2 + \cancel{2} \cdot r_1 + \cancel{2} \cdot r_2 \geq \cancel{3}2.$$

CERTIFIED CORE-GUIDED SEARCH (EXAMPLE)

Objective (*min*): $r_1 + r_2 + r_3 = 1 + p_2 + r_3$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
Core returned by solver:	
$r_1 + r_2 \geq 1$	Reverse Unit Propagation
$2 \cdot \overline{p}_2 + r_1 + r_2 \geq 2$	Fresh variable
$p_2 + \overline{r}_1 + \overline{r}_2 \geq 1$	
$\mathrm{CNF}(p_2 \Leftrightarrow (r_1 + r_2 \geq 2))$	Explicit CP derivation
$r_1 + r_2 = 1 + p_2$	Explicit CP derivation
$\{x_1, x_2, x_3, x_4, r_1, \overline{r}_2, \overline{r}_3\}$	Solution
$\overline{r}_1 + \overline{r}_2 + \overline{r}_3 \geq 3$	Objective Improvement
$0 \geq 1$	Explicit CP derivation

Explicit CP derivations:

CNF encoding (totalizer): cf. LSU

Adding up definition of p_2 and core constraint and dividing by 2 yields

$$\cancel{2} \cdot \overline{p}_2 + \cancel{2} \cdot r_1 + \cancel{2} \cdot r_2 \geq \cancel{3}2.$$

which is the same as $r_1 + r_2 \geq 1 + p_2$.
Other direction already given

CERTIFIED CORE-GUIDED SEARCH (EXAMPLE)

Objective (*min*): $r_1 + r_2 + r_3 = 1 + p_2 + r_3$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
Core returned by solver:	
$r_1 + r_2 \geq 1$	Reverse Unit Propagation
$2 \cdot \overline{p}_2 + r_1 + r_2 \geq 2$	Fresh variable
$p_2 + \overline{r}_1 + \overline{r}_2 \geq 1$	
$\mathrm{CNF}(p_2 \Leftrightarrow (r_1 + r_2 \geq 2))$	Explicit CP derivation
$r_1 + r_2 = 1 + p_2$	Explicit CP derivation
$\{x_1, x_2, x_3, x_4, r_1, \overline{r}_2, \overline{r}_3\}$	Solution
$\overline{r}_1 + \overline{r}_2 + \overline{r}_3 \geq 3$	Objective Improvement
$0 \geq 1$	Explicit CP derivation

Explicit CP derivations:

CNF encoding (totalizer): cf. LSU

Adding up definition of p_2 and core constraint and dividing by 2 yields

$$\cancel{2} \cdot \overline{p}_2 + \cancel{2} \cdot r_1 + \cancel{2} \cdot r_2 \geq \cancel{3}2.$$

which is the same as $r_1 + r_2 \geq 1 + p_2$. Other direction already given

Previously derived cores guarantee that objective is at least 1:
$r_1 + r_2 \ (+ r_3) \geq 1$

CERTIFIED CORE-GUIDED SEARCH (EXAMPLE)

Objective (*min*): $r_1 + r_2 + r_3 = 1 + p_2 + r_3$

VeriPB proof:

derived	justification
$x_2 + r_2 \geq 1$	Reverse Unit Propagation
Core returned by solver:	
$r_1 + r_2 \geq 1$	Reverse Unit Propagation
$2 \cdot \overline{p}_2 + r_1 + r_2 \geq 2$	Fresh variable
$p_2 + \overline{r}_1 + \overline{r}_2 \geq 1$	
CNF$(p_2 \Leftrightarrow (r_1 + r_2 \geq 2))$	Explicit CP derivation
$r_1 + r_2 = 1 + p_2$	Explicit CP derivation
$\{x_1, x_2, x_3, x_4, r_1, \overline{r}_2, \overline{r}_3\}$	Solution
$\overline{r}_1 + \overline{r}_2 + \overline{r}_3 \geq 3$	Objective Improvement
$0 \geq 1$	Explicit CP derivation

Explicit CP derivations:

CNF encoding (totalizer): cf. LSU

Adding up definition of p_2 and core constraint and dividing by 2 yields

$$\cancel{2} \cdot \overline{p}_2 + \cancel{2} \cdot r_1 + \cancel{2} \cdot r_2 \geq \cancel{3}2.$$

which is the same as $r_1 + r_2 \geq 1 + p_2$.
Other direction already given

Previously derived cores guarantee that objective is at least 1:
$r_1 + r_2 \ (+r_3) \geq 1$
Adding this to objective improvement constraint gives contradiction

COMPLETE CG EXAMPLE IN VERIPB SYNTAX

```
pseudo-Boolean proof version 2.0
f 7
* Clauses derived by solver (inc core)
rup 1 x1 1 r2 >= 1 ;
rup 1 r1 1 r2 >= 1 ;
* Introduce fresh variable
red 2 ~p2 1 r1 1 r2  >= 2 ; p2 -> 0 ;
red 1 p2 1 ~r1 1 ~r2 >= 1; p2 -> 1 ;
* Encode this in CNF
pol 10 ~r1 +
pol 10 ~r2 +
* Rewriting the objective
pol 9 10 + 2 d
* Check that we have indeed
*   derived that r1 + r2 = 1 + p2
e 14 : 1 r1 1 r2 -1 p2 >= 1 ;
e 11 : -1 r1 -1 r2 1 p2 >= -1 ;
* Solution found
soli x1 x2 x3 x4 r1 ~r2 ~r3
* Prove optimality of solution:
pol -1 9 +
ia -1 : >= 1 ;
* Conclusion
output NONE
conclusion BOUNDS 1 1
end pseudo-Boolean proof
```

ADVANCED TECHNIQUES FOR CORE-GUIDED MAXSAT

- Important to deal with all state-of-the-art solver techniques

ADVANCED TECHNIQUES FOR CORE-GUIDED MAXSAT

- Important to deal with all state-of-the-art solver techniques
- Additional techniques that are skipped in this example
  - Intrinsic at-most-one constraints [IMM19]

ADVANCED TECHNIQUES FOR CORE-GUIDED MAXSAT

- Important to deal with all state-of-the-art solver techniques
- Additional techniques that are skipped in this example
  - Intrinsic at-most-one constraints [IMM19]
  - Hardening [ABGL12]

ADVANCED TECHNIQUES FOR CORE-GUIDED MAXSAT

- Important to deal with all state-of-the-art solver techniques
- Additional techniques that are skipped in this example
  - Intrinsic at-most-one constraints [IMM19]
  - Hardening [ABGL12]
  - Lazy counter variables [MJML14]

ADVANCED TECHNIQUES FOR CORE-GUIDED MAXSAT

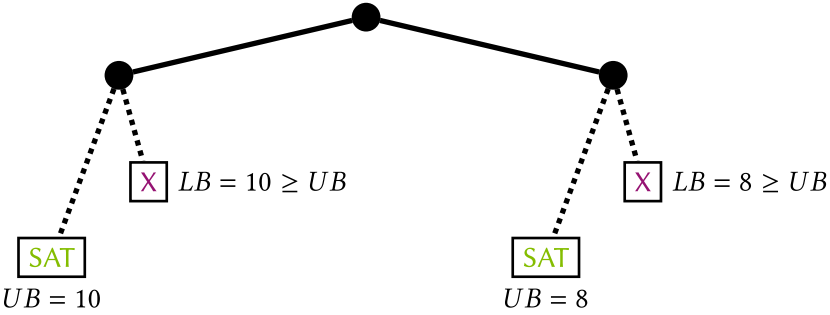
- Important to deal with all state-of-the-art solver techniques
- Additional techniques that are skipped in this example
  - Intrinsic at-most-one constraints [IMM19]
  - Hardening [ABGL12]
  - Lazy counter variables [MJML14]
- VeriPB Proof logging also convenient for these techniques [BBN+23]

OUTLINE

BRANCH AND BOUND

Branch and Bound:

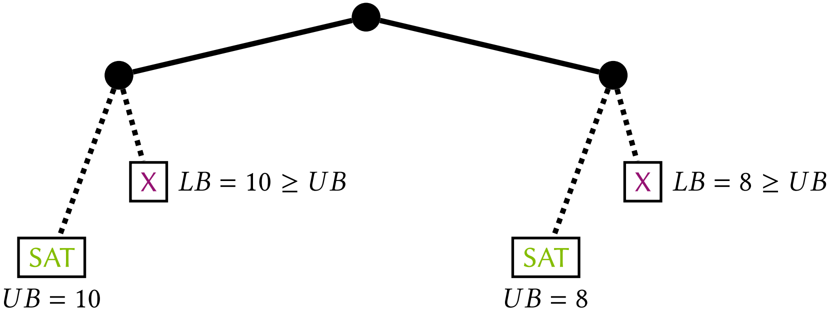
- Explore the search tree for solutions
- Update Upper Bound UB when solution with better objective value is found
- Underestimate Lower Bound LB at every node
- Prune branch when conflict found or when $LB \geq UB$



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)

$f^t = y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 \qquad UB = 3$

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)

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

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)

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

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)

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

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)

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

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)

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

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)

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

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)

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

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:

$\overline{x_2} \wedge \overline{x_4} \wedge \overline{y}_1 \wedge \overline{y}_4 \rightarrow \square$

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

SOFT CONFLICT DETECTION: FULL EXAMPLE (UNWEIGHTED CASE)

$f = y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 \qquad UB = 3$

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: $\overline{x_2} \land \overline{x_4} \rightarrow y_1 \lor y_4$

Core 2: $\overline{x_2} \land x_7 \rightarrow y_2 \lor y_3 \lor y_5$

Core 3: $x_1 \land \overline{x_4} \land x_7 \rightarrow y_6 \lor y_7$

SOFT CONFLICT DETECTION: FULL EXAMPLE (UNWEIGHTED CASE)

$f = y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 \qquad UB = 3$

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: $\overline{x_2} \land \overline{x_4} \rightarrow y_1 \lor y_4$

Core 2: $\overline{x_2} \land x_7 \rightarrow y_2 \lor y_3 \lor y_5$

Core 3: $x_1 \land \overline{x_4} \land x_7 \rightarrow y_6 \lor y_7$

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

SOFT CONFLICT DETECTION: FULL EXAMPLE (UNWEIGHTED CASE)

$f = y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 \qquad UB = 3$

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: $\overline{x_2} \wedge \overline{x_4} \longrightarrow y_1 \vee y_4$

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

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

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

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

SOFT CONFLICT DETECTION: FULL EXAMPLE (UNWEIGHTED CASE)

$f = y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + y_8 \qquad UB = 3$

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: $\overline{x_2} \wedge \overline{x_4} \longrightarrow y_1 \vee y_4$

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

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

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

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

Soft conflict:

$x_1^d\ \overline{x_2}^p\ x_3^p\ \overline{x_4}^d\ x_5^p\ x_6^p\ x_7^p,$ Conflict $\overline{x_1} \vee x_2 \vee x_4 \vee \overline{x_7}$ **(soft conflict)**

SOFT CONFLICT DETECTION (WEIGHTED CASE)

Weighted MaxCDCL

- Weight of Local Core K = smallest weight of objective literals in K
- Each objective literal can contribute to many cores
- The total contribution of a literal cannot exceed its weight

SOFT CONFLICT DETECTION (WEIGHTED CASE)

Weighted MaxCDCL

- Weight of Local Core K = smallest weight of objective literals in K
- Each objective literal can contribute to many cores
- The total contribution of a literal cannot exceed its weight

$f^t = 7y_1 + 2y_2 + 1y_3 + 1y_4 + 1y_5 + 4y_6 + 1y_7 + 3y_8 \qquad UB = 4$

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 K = smallest weight of objective literals in K
- Each objective literal can contribute to many cores
- The total contribution of a literal cannot exceed its weight

$f^t = 7y_1 + 2y_2 + 1y_3 + 1y_4 + 1y_5 + 4y_6 + 1y_7 + 3y_8 \qquad UB = 4$

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 K = smallest weight of objective literals in K
- Each objective literal can contribute to many cores
- The total contribution of a literal cannot exceed its weight

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

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

SOFT CONFLICT DETECTION (WEIGHTED CASE)

Weighted MaxCDCL

- Weight of Local Core K = smallest weight of objective literals in K
- Each objective literal can contribute to many cores
- The total contribution of a literal cannot exceed its weight

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

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 K = smallest weight of objective literals in K
- Each objective literal can contribute to many cores
- The total contribution of a literal cannot exceed its weight

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

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

Core 2: $x_3 \land \overline{x_4} \longrightarrow y_1 \lor y_5$ (weight 1)

SOFT CONFLICT DETECTION (WEIGHTED CASE)

Weighted MaxCDCL

- Weight of Local Core K = smallest weight of objective literals in K
- Each objective literal can contribute to many cores
- The total contribution of a literal cannot exceed its weight

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

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

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

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

SOFT CONFLICT DETECTION (WEIGHTED CASE)

Weighted MaxCDCL

- Weight of Local Core K = smallest weight of objective literals in K
- Each objective literal can contribute to many cores
- The total contribution of a literal cannot exceed its weight

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

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 K = smallest weight of objective literals in K
- Each objective literal can contribute to many cores
- The total contribution of a literal cannot exceed its weight

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

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

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

SOFT CONFLICT DETECTION (WEIGHTED CASE)

Weighted MaxCDCL

- Weight of Local Core K = smallest weight of objective literals in K
- Each objective literal can contribute to many cores
- The total contribution of a literal cannot exceed its weight

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

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

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

Conclusion: $x_1 \wedge \overline{x_2} \wedge \overline{x_4} \longrightarrow LB = 5 \geq 4 = UB$ **Soft Conflict clause**: $\overline{x}_1 \vee x_2 \vee x_4$

PROOF LOGGING SOFT CONFLICTS

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

PROOF LOGGING SOFT CONFLICTS

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

Found "disjoint" cores

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

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

PROOF LOGGING SOFT CONFLICTS

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

Found "disjoint" cores

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

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

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

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

PROOF LOGGING SOFT CONFLICTS

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

Found "disjoint" cores (RUP)

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

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

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

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

PROOF LOGGING SOFT CONFLICTS

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

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

Found "disjoint" cores (RUP)

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

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

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

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

Multiplication by their weight 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: $\overline{x}_1 + x_2 + x_4 \geq 1 \qquad UB = 4$

Found "disjoint" cores (RUP)

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

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

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

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

Multiplication by their weight 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: $\overline{x}_1 + x_2 + x_4 \geq 1 \qquad UB = 4$

Found "disjoint" cores (RUP)

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

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

Core 2: $x_1 \longrightarrow y_1 \lor y_6 \lor y_8$ (3)

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

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

Found "disjoint" cores (RUP)

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

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

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

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

Multiplication by their weight 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: $\overline{x}_1 + x_2 + x_4 \geq 1 \qquad UB = 4$

Found "disjoint" cores (RUP)

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

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

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

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

Multiplication by their weight 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: $\overline{x}_1 + x_2 + x_4 \geq 1 \qquad UB = 4$

Found "disjoint" cores (RUP)

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

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

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

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

Multiplication by their weight 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: $\overline{x}_1 + x_2 + x_4 \geq 1 \qquad UB = 4$

Found "disjoint" cores (RUP)

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

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

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

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

Multiplication by their weight 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

Certifying Optimality:

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

PROOF LOGGING MAXCDCL

Certifying Optimality:

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

Extra techniques included in paper:

- Literal Unlocking for unweighted case
  - Find cardinality constraints over disjoint set literals as “local cores”
- Encoding Solution-Improving Constraint using Multi-Valued Decision Diagram encoding

OUTLINE

SUMMING UP

- Combinatorial solving and optimization is a true success story
- But ensuring correctness is a crucial, and not yet satisfactorily addressed, concern
- Certifying solvers producing machine-verifiable proofs of correctness seems like most promising approach
- Cutting planes reasoning with pseudo-Boolean constraints seems to hit a sweet spot between simplicity and expressivity
- Here demonstrated for MaxSAT, but also used in many other applications

CERTIFIED FIRST-ORDER MODEL EXPANSION (CERTIFOX)

- Start from first-order problem representation

CERTIFIED FIRST-ORDER MODEL EXPANSION (CERTIFOX)

- Start from first-order problem representation
- Study various forms of proof composition and without-loss-of-generality reasoning

CERTIFIED FIRST-ORDER MODEL EXPANSION (CERTIFOX)

- Start from first-order problem representation
- Study various forms of proof composition and without-loss-of-generality reasoning
- Interested? I'm looking for PostDocs to join the proof logging revolution.

https://www.bartbogaerts.eu/projects/CertiFOX/

REFERENCES

[ABGL12] Carlos Ansótegui, María Luisa Bonet, Joel Gabàs, and Jordi Levy. Improving SAT-based weighted MaxSAT solvers. In *Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming (CP '12)*, volume 7514 of *Lecture Notes in Computer Science*, pages 86–101. Springer, October 2012.

[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.

[ANOR09] Roberto Asín, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell. Cardinality networks and their applications. In Oliver Kullmann, editor, *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, volume 5584 of *Lecture Notes in Computer Science*, pages 167–180. Springer, 2009.

[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.

[Bar95] Peter Barth. A Davis-Putnam based enumeration algorithm for linear pseudo-Boolean optimization. Technical Report MPI-I-95-2-003, Max-Planck-Institut für Informatik, January 1995.

REFERENCES

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

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

[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 *CP*, volume 307 of *LIPIcs*, pages 4:1–4:28. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

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

[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.

[Bla37] Archie Blake. *Canonical Expressions in Boolean Algebra*. PhD thesis, University of Chicago, 1937.

REFERENCES

[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.

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

[BMN22] Bart Bogaerts, Ciaran McCreesh, and Jakob Nordström. Solving with provably correct results: Beyond satisfiability, and towards constraint programming. Tutorial at the *28th International Conference on Principles and Practice of Constraint Programming*. Slides available at `http://www.jakobnordstrom.se/presentations/`, August 2022.

[BN21] Samuel R. Buss and Jakob Nordström. Proof complexity and SAT solving. In Biere et al. [BHvMW21], chapter 7, pages 233–350.

[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.

[BS97] Roberto J. Bayardo Jr. and Robert Schrag. Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI '97)*, pages 203–208, July 1997.

REFERENCES

[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.

[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.

[DLL62] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, July 1962.

[DP60] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, 1960.

REFERENCES

[EG21] Leon Eifler and Ambros Gleixner. A computational status update for exact rational mixed integer programming. In *Proceedings of the 22nd International Conference on Integer Programming and Combinatorial Optimization (IPCO '21)*, volume 12707 of *Lecture Notes in Computer Science*, pages 163–177. Springer, May 2021.

[EGMN20] Jan Elffers, Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. Justifying all differences using pseudo-Boolean reasoning. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI '20)*, pages 1486–1494, February 2020.

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

[GMM+20] Stephan Gocht, Ross McBride, Ciaran McCreesh, Jakob Nordström, Patrick Prosser, and James Trimble. Certifying solvers for clique and maximum common (connected) subgraph problems. In *Proceedings of the 26th International Conference on Principles and Practice of Constraint Programming (CP '20)*, volume 12333 of *Lecture Notes in Computer Science*, pages 338–357. Springer, September 2020.

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

REFERENCES

[GMN22] Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. An auditable constraint programming solver. In *Proceedings of the 28th International Conference on Principles and Practice of Constraint Programming (CP '22)*, volume 235 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:18, August 2022.

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

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

[GN21] Stephan Gocht and Jakob Nordström. Certifying parity reasoning efficiently using pseudo-Boolean proofs. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI '21)*, pages 3768–3777, February 2021.

[Goc22] Stephan Gocht. *Certifying Correctness for Combinatorial Algorithms by Using Pseudo-Boolean Reasoning*. PhD thesis, Lund University, June 2022. Available at `https://portal.research.lu.se/en/publications/certifying-correctness-for-combinatorial-algorithms-by-using-pseu`.

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

REFERENCES

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

[HHW13a] Marijn J. H. Heule, Warren A. Hunt Jr., and Nathan Wetzler. Trimming while checking clausal proofs. In *Proceedings of the 13th International Conference on Formal Methods in Computer-Aided Design (FMCAD '13)*, pages 181–188, October 2013.

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

[IMM19] Alexey Ignatiev, António Morgado, and João Marques-Silva. RC2: an efficient maxsat solver. *J. Satisf. Boolean Model. Comput.*, 11(1):53–64, 2019.

[KM21] Sonja Kraiczy and Ciaran McCreesh. Solving graph homomorphism and subgraph isomorphism problems faster through clique neighbourhood constraints. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI '21)*, pages 1396–1402, August 2021.

[MJML14] Ruben Martins, Saurabh Joshi, Vasco M. Manquinho, and Inês Lynce. Incremental cardinality constraints for MaxSAT. In *Proceedings of the 20th International Conference on Principles and Practice of Constraint Programming (CP '14)*, volume 8656 of *Lecture Notes in Computer Science*, pages 531–548. Springer, September 2014.

REFERENCES

[MM23] Matthew McIlree and Ciaran McCreesh. Proof logging for smart extensional constraints. In *Proceedings of the 29th International Conference on Principles and Practice of Constraint Programming (CP '23)*, volume 280 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:17, August 2023.

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

[MMZ+01] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference (DAC '01)*, pages 530–535, June 2001.

[MS99] João P. Marques-Silva and Karem A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, May 1999. Preliminary version in *ICCAD '96*.

[OLH+13] Toru Ogawa, Yangyang Liu, Ryuzo Hasegawa, Miyuki Koshimura, and Hiroshi Fujita. Modulo based CNF encoding of cardinality constraints and its application to maxsat solvers. In *25th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2013, Herndon, VA, USA, November 4-6, 2013*, pages 9–17. IEEE Computer Society, 2013.

[PRB18] Tobias Paxian, Sven Reimer, and Bernd Becker. Dynamic polynomial watchdog encoding for solving weighted MaxSAT. In *Proceedings of the 21st International Conference on Theory and Applications of Satisfiability Testing (SAT '18)*, volume 10929 of *Lecture Notes in Computer Science*, pages 37–53. Springer, July 2018.

REFERENCES

[RM16] Olivier Roussel and Vasco M. Manquinho. Input/output format and solver requirements for the competitions of pseudo-Boolean solvers. Revision 2324. Available at `http://www.cril.univ-artois.fr/PB16/format.pdf`, January 2016.

[Rob65] John Alan Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, January 1965.

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

[Tse68] Grigori Tseitin. On the complexity of derivation in propositional calculus. In A. O. Silenko, editor, *Structures in Constructive Mathematics and Mathematical Logic, Part II*, pages 115–125. Consultants Bureau, New York-London, 1968.

[Van08] Allen Van Gelder. Verifying RUP proofs of propositional unsatisfiability. In *10th International Symposium on Artificial Intelligence and Mathematics (ISAIM '08)*, 2008. Available at `http://isaim2008.unl.edu/index.php?page=proceedings`.

[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, 2023. To appear.

REFERENCES

[VDB22] Dieter Vandesande, Wolf De Wulf, and Bart Bogaerts. QMaxSATpb: A certified MaxSAT solver. In *Proceedings of the 16th International Conference on Logic Programming and Non-monotonic Reasoning (LPNMR '22)*, volume 13416 of *Lecture Notes in Computer Science*, pages 429–442. Springer, September 2022.

[War98] Joost P. Warners. A linear-time transformation of linear inequalities into conjunctive normal form. *Information Processing Letters*, 68(2):63–69, October 1998.

[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.

ACKNOWLEDGEMENTS

This work was partially supported by Fonds Wetenschappelijk Onderzoek – Vlaanderen (project G070521N).

Co-funded by the European Union (ERC, CertiFOX, 101122653). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.