Certifying Automated Reasoning

Jeremias Berg

Department of Computer Science, University of Helsinki, Finland

Helsinki Algorithms and Theory Days
August 30

Joint work with: Matti Järvisalo, Hannes Ihalainen, Christoph Jabs,
Bart Bogaerts, Jakob Nordström, Andy Oertel, Yong Kiam Tan,
Dieter Vandesande, Magnus Myreen

Automated reasoning

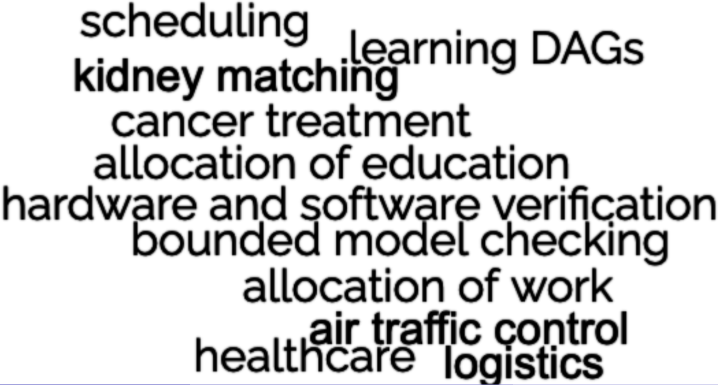
Significant progress in last couple of decades on combinatorial solvers

- Boolean satisfiability (SAT) & modulo theories (SMT), solving and optimization [Biere, Heule, van Maaren, and Walsh, 2021]
- Constraint programming [Rossi, van Beek, and Walsh, 2006]
- Pseudo-boolean (0-1 integer linear programming) [Elffers and Nordström, 2020].

Automated reasoning

Significant progress in last couple of decades on combinatorial solvers

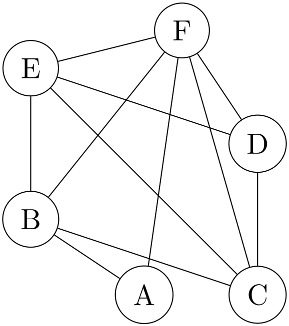
- Boolean satisfiability (SAT) & modulo theories (SMT), solving and optimization [Biere, Heule, van Maaren, and Walsh, 2021]
- Constraint programming [Rossi, van Beek, and Walsh, 2006]
- Pseudo-boolean (0-1 integer linear programming) [Elffers and Nordström, 2020].



Example problem

Maximum Clique

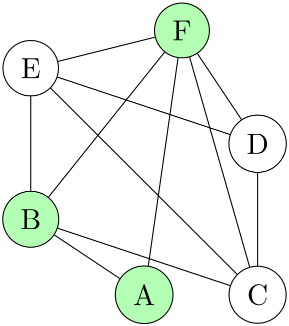
Decision problem: Is there clique of size 3?



Example problem

Maximum Clique

Decision problem: Is there clique of size 3? yes

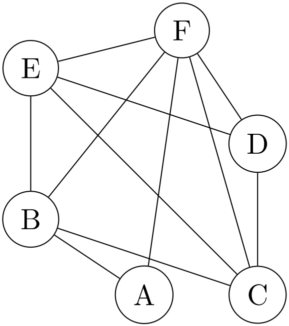


Example problem

Maximum Clique

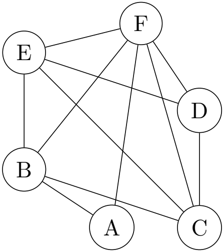
Decision problem: Is there clique of size 3? yes

Optimization problem: What is the size of the largest clique?



Automated Reasoning

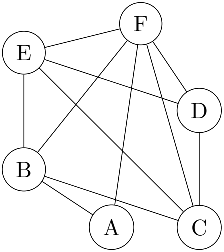
for solving maximum clique



Problem instance

Automated Reasoning

for solving maximum clique



Maximize: $x_A + x_B + x_C + x_D + x_E + x_F$

subject to:

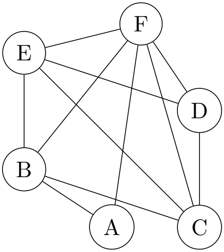
$$(1 - x_A) + (1 - x_C) \geq 1$$
$$(1 - x_D) + (1 - x_B) \geq 1$$
$$\vdots$$
$$x_j \in \{0, 1\} \quad \forall j$$

$(1 - x) \equiv \neg x$

Problem instance → Constraint encoding

Automated Reasoning

for solving maximum clique



Maximize: $x_A + x_B + x_C + x_D + x_E + x_F$

subject to:

$$(1 - x_A) + (1 - x_C) \geq 1$$
$$(1 - x_D) + (1 - x_B) \geq 1$$
$$\vdots$$
$$x_j \in \{0, 1\} \quad \forall j$$

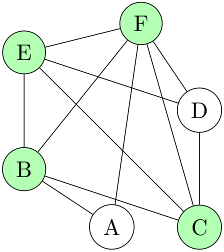
$(1 - x) \equiv \neg x$

Problem instance → Constraint encoding → Solution to constraints

Green=1, Red=0

Automated Reasoning

for solving maximum clique



Maximize: $x_A + x_B + x_C + x_D + x_E + x_F$

subject to:

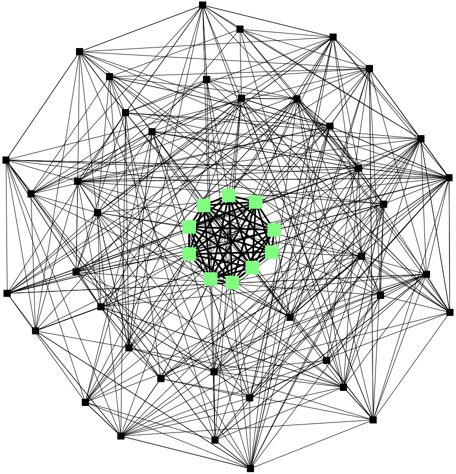
$$(1 - x_A) + (1 - x_C) \geq 1$$
$$(1 - x_D) + (1 - x_B) \geq 1$$
$$\vdots$$
$$x_j \in \{0, 1\} \quad \forall j$$

$(1 - x) \equiv \neg x$

Problem instance → Constraint encoding → Solution to constraints (Green=1, Red=0) → Solution to problem

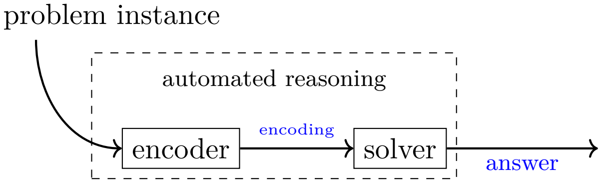
The main question of the day

Can we trust the answer?



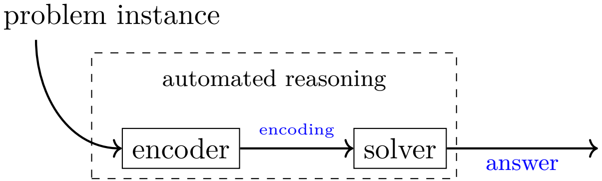
Automated reasoning

in general



Automated reasoning

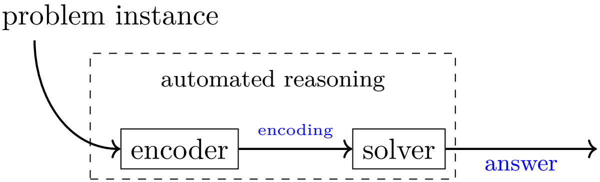
in general



3 main approaches toward trustworthiness:

Automated reasoning

in general

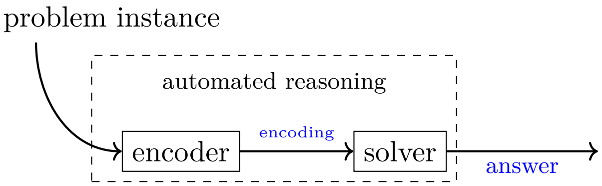


3 main approaches toward trustworthiness:

testing

Automated reasoning

in general



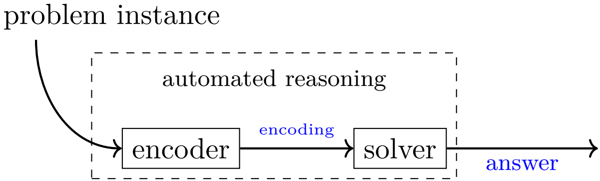
3 main approaches toward trustworthiness:

testing

formal verification

Automated reasoning

in general

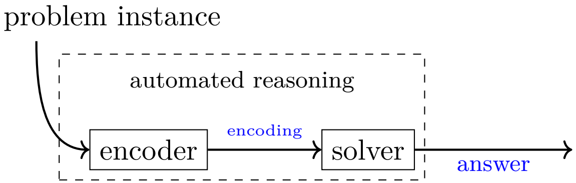


3 main approaches toward trustworthiness:

testing | formal verification | proof logging

Proof Logging

[Järvisalo, Heule, and Biere, 2012; Wetzler, Heule, and Jr., 2014; Heule, 2021; van Doornmalen, Eifler, Gleixner, and Hojny, 2023; Bogaerts, Gocht, McCreesh, and Nordström, 2022]

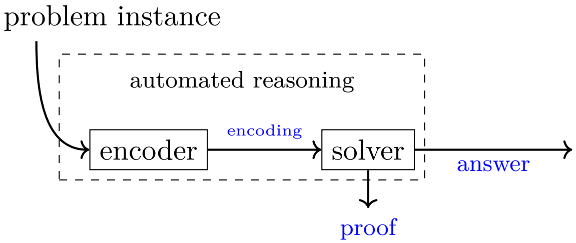


Desiderata of proof format

- powerful
- simple

Proof Logging

[Järvisalo, Heule, and Biere, 2012; Wetzler, Heule, and Jr., 2014; Heule, 2021; van Doornmalen, Eifler, Gleixner, and Hojny, 2023; Bogaerts, Gocht, McCreesh, and Nordström, 2022]

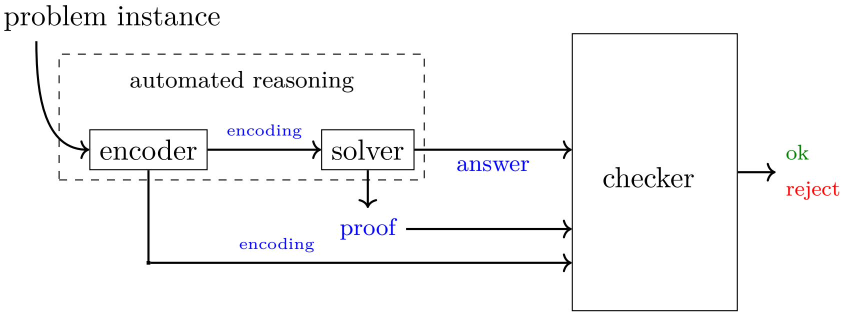


Desiderata of proof format

- powerful
- simple

Proof Logging

[Järvisalo, Heule, and Biere, 2012; Wetzler, Heule, and Jr., 2014; Heule, 2021; van Doornmalen, Eifler, Gleixner, and Hojny, 2023; Bogaerts, Gocht, McCreesh, and Nordström, 2022]

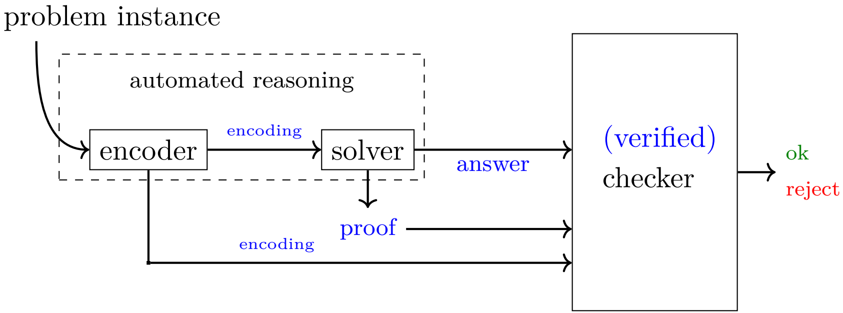


Desiderata of proof format

- powerful
- simple

Proof Logging

[Järvisalo, Heule, and Biere, 2012; Wetzler, Heule, and Jr., 2014; Heule, 2021; van Doornmalen, Eifler, Gleixner, and Hojny, 2023; Bogaerts, Gocht, McCreesh, and Nordström, 2022]

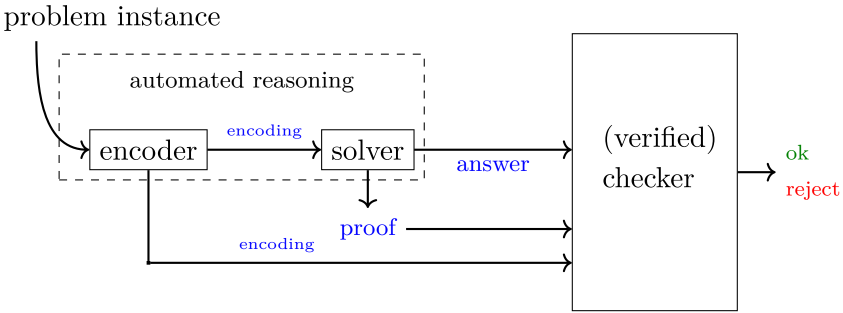


Desiderata of proof format

- powerful
- simple

Proof Logging

[Järvisalo, Heule, and Biere, 2012; Wetzler, Heule, and Jr., 2014; Heule, 2021; van Doornmalen, Eifler, Gleixner, and Hojny, 2023; Bogaerts, Gocht, McCreesh, and Nordström, 2022]

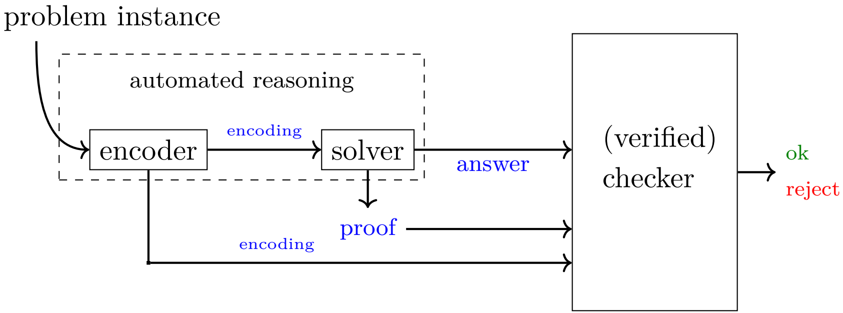


Desiderata of proof format

- powerful
- simple

Proof Logging

[Järvisalo, Heule, and Biere, 2012; Wetzler, Heule, and Jr., 2014; Heule, 2021; van Doornmalen, Eifler, Gleixner, and Hojny, 2023; Bogaerts, Gocht, McCreesh, and Nordström, 2022]

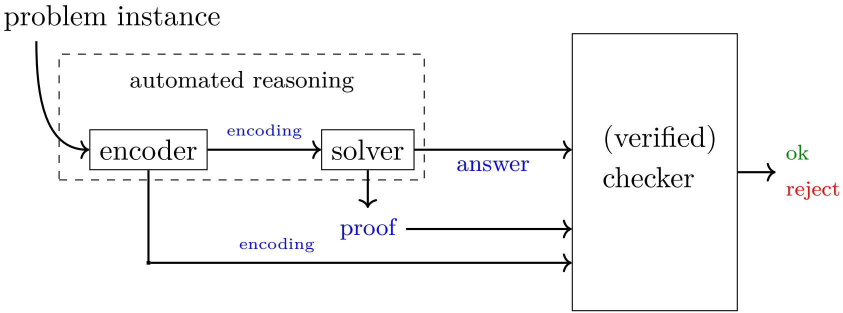


Desiderata of proof format

- powerful
- simple

Proof Logging

[Järvisalo, Heule, and Biere, 2012; Wetzler, Heule, and Jr., 2014; Heule, 2021; van Doornmalen, Eifler, Gleixner, and Hojny, 2023; Bogaerts, Gocht, McCreesh, and Nordström, 2022]



Desiderata of proof format

- powerful
- simple

Redundance-based proofs

Concrete Constraints

Propositional Logic, SAT, MaxSAT

- Instance:
  - Set of clauses, (CNF formula)
  - a linear objective function cost
- Find assignment τ that:
  - satisfies all clauses and
  - minimizes cost

Concrete Constraints

Propositional Logic, SAT, MaxSAT

- Instance:
  - Set of clauses, (CNF formula)
  - a linear objective function cost
- Find assignment τ that:
  - satisfies all clauses and
  - minimizes cost

$$F = \{(b_1 \vee x), (\neg x \vee b_2), (b_2 \vee y), (\neg y, b_3)\}$$

$$\text{cost} \equiv 2b_1 + 4b_2 + b_3$$

Concrete Constraints

Propositional Logic, SAT, MaxSAT

- Instance:
  - Set of clauses, (CNF formula)
  - a linear objective function cost
- Find assignment τ that:
  - satisfies all clauses and
  - minimizes cost

$$\tau(\mathrm{y}) = \tau(\mathrm{b}_1) = \tau(\mathrm{b}_3) = 1$$
$$\tau(\mathrm{x}) = \tau(\mathrm{b}_2) = 0$$

$$\mathrm{F} = \{(\mathrm{b}_1 \vee \mathrm{x}), (\neg \mathrm{x} \vee \mathrm{b}_2), (\mathrm{b}_2 \vee \mathrm{y}), (\neg \mathrm{y}, \mathrm{b}_3)\}$$

$$\mathrm{cost} \equiv 2\mathrm{b}_1 + 4\mathrm{b}_2 + \mathrm{b}_3$$

$$\mathrm{cost}(\tau) = 3$$

Clause Redundancy

[Järvisalo, Heule, and Biere, 2012; Heule, Kiesl, and Biere, 2020; Ihalainen, Berg, and Järvisalo, 2022]

Definition

Clause C is redundant for formula F and objective cost if

$$\text{minimum-cost}(F) = \text{minimum-cost}(F \wedge C)$$

(wrt. cost)

equisatisfiability a special case

Example:

$(x \vee b_1) \wedge (\neg x \vee b_2)$

$\text{cost} = b_1 + 2b_2$

Clause Redundancy

[Järvisalo, Heule, and Biere, 2012; Heule, Kiesl, and Biere, 2020; Ihalainen, Berg, and Järvisalo, 2022]

Definition

Clause C is redundant for formula F and objective cost if

$$\text{minimum-cost}(F) = \text{minimum-cost}(F \wedge C)$$

(wrt. cost)

equisatisfiability a special case

Example:

$(x \vee b_1) \wedge (\neg x \vee b_2)$

$\text{cost} = b_1 + 2b_2$

Clause Redundancy

[Järvisalo, Heule, and Biere, 2012; Heule, Kiesl, and Biere, 2020; Ihalainen, Berg, and Järvisalo, 2022]

Definition

Clause C is redundant for formula F and objective cost if

$$\text{minimum-cost}(F) = \text{minimum-cost}(F \wedge C)$$

(wrt. cost)

equisatisfiability a special case

Example:

$(x \vee b_1) \wedge (\neg x \vee b_2)$ $\quad$ $(\neg b_2)$ is redundant

$\text{cost} = b_1 + 2b_2$

Clause Redundancy

[Järvisalo, Heule, and Biere, 2012; Heule, Kiesl, and Biere, 2020; Ihalainen, Berg, and Järvisalo, 2022]

Definition

Clause C is redundant for formula F and objective cost if

$$\text{minimum-cost}(F) = \text{minimum-cost}(F \land C)$$

(wrt. cost)

equisatisfiability a special case

Example:

$(x \lor b_1) \land (\neg x \lor b_2)$

$\text{cost} = b_1 + 2b_2$

$(\neg b_2)$ is redundant

$(\neg b_1)$ is not redundant

Characterizing redundancy

[Heule, Kiesl, and Biere, 2020; Ihalainen, Berg, and Järvisalo, 2022]

(informal) Theorem

C redundant for F and cost iff there exists a set of literals L_C that fixes any solution τ of F that falsifies C without increasing its cost.

Example

$F = (x \lor b_1) \land (\neg x \lor b_2 \lor b_1)$

$\text{cost} = b_1 + 2b_2$

$C = (\neg b_2)$ is redundant

$L_C = \{\neg b_2, b_1\}$

Characterizing redundancy

[Heule, Kiesl, and Biere, 2020; Ihalainen, Berg, and Järvisalo, 2022]

(informal) Theorem

C redundant for F and cost iff there exists a set of literals L_C that fixes any solution τ of F that falsifies C without increasing its cost.

Example

$F = (x \vee b_1) \wedge (\neg x \vee b_2 \vee b_1)$

$\mathrm{cost} = b_1 + 2b_2$

$C = (\neg b_2)$ is redundant

$L_C = \{\neg b_2, b_1\}$

Characterizing redundancy

[Heule, Kiesl, and Biere, 2020; Ihalainen, Berg, and Järvisalo, 2022]

(informal) Theorem

C redundant for F and cost iff there exists a set of literals L_C that fixes any solution τ of F that falsifies C without increasing its cost.

Example

$$F = (x \lor b_1) \land (\neg x \lor b_2 \lor b_1)$$

$$\text{cost} = b_1 + 2b_2$$

$C = (\neg b_2)$ is redundant

$$L_C = \{\neg b_2, b_1\}$$

If τ satisfies F but falsifies C then assigning $b_1 = 1$, $b_2 = 0$ and the rest according to τ satisfies $F \land C$ with cost less than τ.

Characterizing redundancy

[Heule, Kiesl, and Biere, 2020; Ihalainen, Berg, and Järvisalo, 2022]

(informal) Theorem

C redundant for F and cost iff there exists a set of literals L_C that fixes any solution τ of F that falsifies C without increasing its cost.

Example

$F = (x \vee b_1) \wedge (\neg x \vee b_2 \vee b_1) \wedge F' \longleftarrow$ F' does not contain $b_1, b_2, \neg b_1$ or $\neg b_2$

$\text{cost} = b_1 + 2b_2$

$C = (\neg b_2)$ is redundant

$L_C = \{\neg b_2, b_1\}$

Note: adding redundant clauses might change the set of solutions

A (very simplified) redundancy-based proof

e.g. [Heule, Kiesl, and Biere, 2020; Bogaerts, Gocht, McCreesh, and Nordström, 2022]

A proof for $F = \{C_1, \ldots, C_n\}$ and cost is a sequence:

$$C_1, C_2, \ldots, C_n, C_{n+1}, \ldots [] \text{ =empty clause}$$

s.t. each C_{n+t} is either:

- redundant wrt. $C_1 \wedge \ldots \wedge C_{n+t-1}$, or
- cost $< \text{cost}(\tau)$ for a solution τ of $C_1 \wedge \ldots \wedge C_{n+t-1}$.

A (very simplified) redundancy-based proof

e.g. [Heule, Kiesl, and Biere, 2020; Bogaerts, Gocht, McCreesh, and Nordström, 2022]

A proof for $F = \{C_1, \ldots, C_n\}$ and cost is a sequence:

$$C_1, C_2, \ldots, C_n, C_{n+1}, \ldots [] \text{ =empty clause}$$

s.t. each C_{n+t} is either:

- redundant wrt. $C_1 \wedge \ldots \wedge C_{n+t-1}$, or
- $cost < cost(\tau)$ for a solution τ of $C_1 \wedge \ldots \wedge C_{n+t-1}$.

The proof establishes:

- optimality if $C_{n+t} = cost < cost(\tau)$ for some t
- infeasibility of constraints, otherwise

A (very simplified) redundancy-based proof

e.g. [Heule, Kiesl, and Biere, 2020; Bogaerts, Gocht, McCreesh, and Nordström, 2022]

A proof for $F = \{C_1, \ldots, C_n\}$ and cost is a sequence:

$$C_1, C_2, \ldots, C_n, C_{n+1}, \ldots [] \text{ =empty clause}$$

s.t. each C_{n+t} is either:

- redundant wrt. $C_1 \wedge \ldots \wedge C_{n+t-1}$, or
- $\text{cost} < \text{cost}(\tau)$ for a solution τ of $C_1 \wedge \ldots \wedge C_{n+t-1}$.

The proof establishes:

- optimality if $C_{n+t} = \text{cost} < \text{cost}(\tau)$ for some t
- infeasibility of constraints, otherwise

redundancy-based proof systems are strong

A (very simplified) redundancy-based proof

e.g. [Heule, Kiesl, and Biere, 2020; Bogaerts, Gocht, McCreesh, and Nordström, 2022]

A proof for $F = \{C_1, \ldots, C_n\}$ and cost is a sequence:

$$C_1, C_2, \ldots, C_n, C_{n+1}, \ldots [] \text{ =empty clause}$$

s.t. each C_{n+t} is either:

- redundant wrt. $C_1 \wedge \ldots \wedge C_{n+t-1}$, or
- $\text{cost} < \text{cost}(\tau)$ for a solution τ of $C_1 \wedge \ldots \wedge C_{n+t-1}$.

The proof establishes:

- optimality if $C_{n+t} = \text{cost} < \text{cost}(\tau)$ for some t
- infeasibility of constraints, otherwise

redundancy-based proof systems are strong
need to be careful with deletion

Redunancy for simulating solver reasoning

Subsumed Literal Elimination

[Berg, Saikko, and Järvisalo, 2016; Korhonen, Berg, Saikko, and Järvisalo, 2017]

Assume:
i) b_2 appears at least in the same clauses as b_1.
ii) the coefficient of b_2 in cost is at most the coefficient of b_1.
Then fix $b_2 = 0$ and simplify.

Reasoning

$$\text{cost} = b_1 + 2b_2$$

$$(x \lor b_1) \land (\neg x \lor b_2 \lor b_1)$$

Redunancy for simulating solver reasoning

Subsumed Literal Elimination

[Berg, Saikko, and Järvisalo, 2016; Korhonen, Berg, Saikko, and Järvisalo, 2017]

Assume:

i) b_2 appears at least in the same clauses as b_1.

ii) the coefficient of b_2 in cost is at most the coefficient of b_1.

Then fix $b_2 = 0$ and simplify.

Reasoning

$$\text{cost} = b_1 + 2b_2$$

$$(x \lor b_1) \land (\neg x \lor b_2 \lor b_1)$$

$$\downarrow$$

$$(x \lor b_1) \land (\neg x \lor b_1)$$

Redunancy for simulating solver reasoning

Subsumed Literal Elimination

[Berg, Saikko, and Järvisalo, 2016; Korhonen, Berg, Saikko, and Järvisalo, 2017]

Assume:

i) b_2 appears at least in the same clauses as b_1.

ii) the coefficient of b_2 in cost is at most the coefficient of b_1.

Then fix $b_2 = 0$ and simplify.

Reasoning

$\text{cost} = b_1 + 2b_2$

$(x \lor b_1) \land (\neg x \lor b_2 \lor b_1)$

$\downarrow$

$(x \lor b_1) \land (\neg x \lor b_1)$

$\downarrow$

(b_1)

Redunancy for simulating solver reasoning

Subsumed Literal Elimination

[Berg, Saikko, and Järvisalo, 2016; Korhonen, Berg, Saikko, and Järvisalo, 2017]

Assume:
i) b_2 appears at least in the same clauses as b_1.
ii) the coefficient of b_2 in cost is at most the coefficient of b_1.
Then fix $b_2 = 0$ and simplify.

Reasoning

$\mathrm{cost} = b_1 + 2b_2$

$(x \lor b_1) \land (\neg x \lor b_2 \lor b_1)$

$\downarrow$

$(x \lor b_1) \land (\neg x \lor b_1)$

$\downarrow$

(b_1)

$\downarrow$

$\emptyset$

Redunancy for simulating solver reasoning

Subsumed Literal Elimination

[Berg, Saikko, and Järvisalo, 2016; Korhonen, Berg, Saikko, and Järvisalo, 2017]

Assume:

i) b_2 appears at least in the same clauses as b_1.

ii) the coefficient of b_2 in cost is at most the coefficient of b_1.

Then fix $b_2 = 0$ and simplify.

Reasoning

$\text{cost} = b_1 + 2b_2$

$(x \lor b_1) \land (\neg x \lor b_2 \lor b_1)$

$\downarrow$

$(x \lor b_1) \land (\neg x \lor b_1)$

$\downarrow$

(b_1)

$\downarrow$

$\emptyset$

Redundancy

$(x \lor b_1) \land (\neg x \lor b_2 \lor b_1)$

Redunancy for simulating solver reasoning

Subsumed Literal Elimination

[Berg, Saikko, and Järvisalo, 2016; Korhonen, Berg, Saikko, and Järvisalo, 2017]

Assume:
i) b_2 appears at least in the same clauses as b_1.
ii) the coefficient of b_2 in cost is at most the coefficient of b_1.
Then fix $b_2 = 0$ and simplify.

Reasoning

$\text{cost} = b_1 + 2b_2$

$(x \lor b_1) \land (\neg x \lor b_2 \lor b_1)$

$\downarrow$

$(x \lor b_1) \land (\neg x \lor b_1)$

$\downarrow$

(b_1)

$\downarrow$

$\emptyset$

Redundancy

$(x \lor b_1) \land (\neg x \lor b_2 \lor b_1)$

$\downarrow$

add $(\neg b_2)$

Redunancy for simulating solver reasoning

Subsumed Literal Elimination

[Berg, Saikko, and Järvisalo, 2016; Korhonen, Berg, Saikko, and Järvisalo, 2017]

Assume:
i) b_2 appears at least in the same clauses as b_1.
ii) the coefficient of b_2 in cost is at most the coefficient of b_1.
Then fix $b_2 = 0$ and simplify.

Reasoning

$\text{cost} = b_1 + 2b_2$

$(x \lor b_1) \land (\neg x \lor b_2 \lor b_1)$

$\downarrow$

$(x \lor b_1) \land (\neg x \lor b_1)$

$\downarrow$

(b_1)

$\downarrow$

$\emptyset$

Redundancy

$(x \lor b_1) \land (\neg x \lor b_2 \lor b_1)$

$\downarrow$

add $(\neg b_2)$

$\downarrow$

add (b_1)

Redunancy for simulating solver reasoning

Subsumed Literal Elimination

[Berg, Saikko, and Järvisalo, 2016; Korhonen, Berg, Saikko, and Järvisalo, 2017]

Assume:

i) b_2 appears at least in the same clauses as b_1.

ii) the coefficient of b_2 in cost is at most the coefficient of b_1.

Then fix $b_2 = 0$ and simplify.

Reasoning

$\text{cost} = b_1 + 2b_2$

$(x \lor b_1) \land (\neg x \lor b_2 \lor b_1)$

$\downarrow$

$(x \lor b_1) \land (\neg x \lor b_1)$

$\downarrow$

(b_1)

$\downarrow$

$\emptyset$

Redundancy

$(x \lor b_1) \land (\neg x \lor b_2 \lor b_1)$

$\downarrow$

add $(\neg b_2)$

$\downarrow$

add (b_1)

$\downarrow$

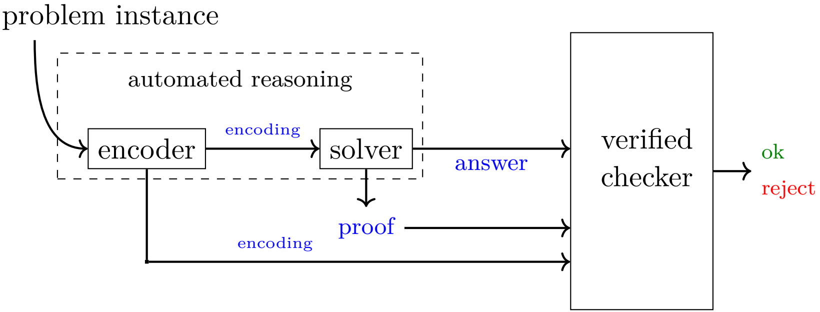
remove $(\neg x \lor b_1 \lor b_2)$

$\downarrow$

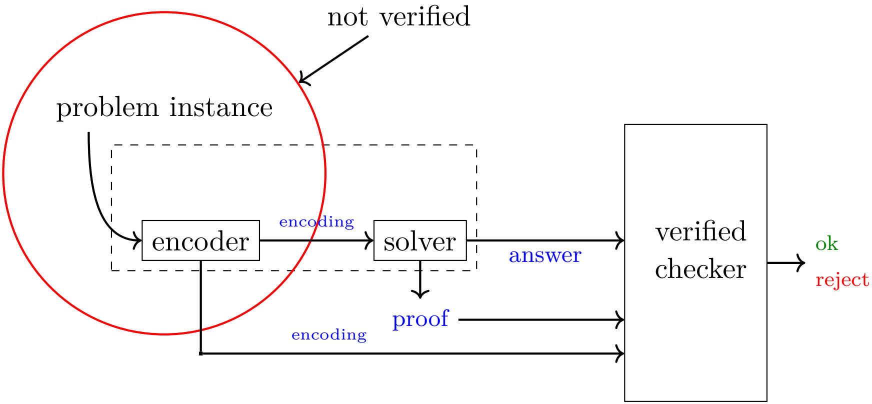
.

What do we need to trust?

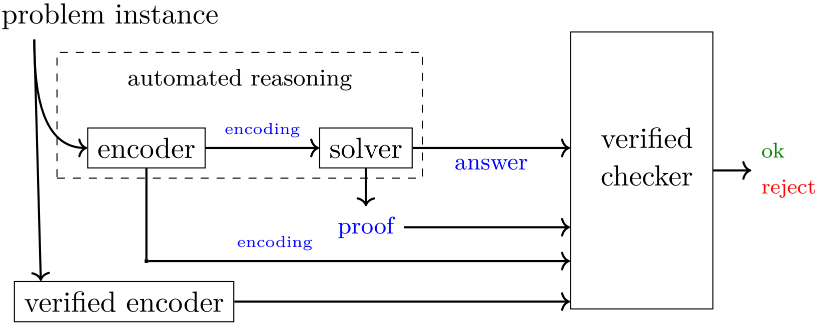
Recap: Certified Automated Reasoning



What about the encoding?



What about the encoding?



- problem-specific verified encoder can prove the right properties of the encoding

What are these right properties?

[Gocht, McCreesh, Myreen, Nordström, Oertel, and Tan, 2024; Ihalainen, Oertel, Tan, Berg, Järvisalo, Myreen, and Nordström, 2024]

$$
\begin{aligned}
&\mathsf{is\_clique}\ vs\ (v,e) \stackrel{\mathsf{def}}{=} \\
&\quad vs \subseteq \{\ 0,1,\ldots,v-1\ \} \land \\
&\quad \forall\, x\ y.\ x \in vs \land y \in vs \land x \neq y \Rightarrow \mathsf{is\_edge}\ e\ x\ y \\
&\mathsf{max\_clique\_size}\ g \stackrel{\mathsf{def}}{=} \mathsf{max}_{\mathsf{set}}\ \{\ \mathsf{card}\ vs \mid \mathsf{is\_clique}\ vs\ g\ \}
\end{aligned}
$$

What are we trusting now?

- e.g. HOL model of verified checkers and correspondence to real system
- HOL4 theorem prover, including logic, implementation, and execution environment [Slind and Norrish, 2008]

What are these right properties?

[Gocht, McCreesh, Myreen, Nordström, Oertel, and Tan, 2024; Ihalainen, Oertel, Tan, Berg, Järvisalo, Myreen, and Nordström, 2024]

$$\begin{aligned}
&\textsf{is_clique}\ vs\ (v,e) \stackrel{\text{def}}{=} \\
&\quad vs \subseteq \{\ 0,1,\ldots,v-1\ \} \land \\
&\quad \forall\, x\ y.\ x \in vs \land y \in vs \land x \neq y \Rightarrow \textsf{is_edge}\ e\ x\ y \\
&\textsf{max_clique_size}\ g \stackrel{\text{def}}{=} \textsf{max}_{\textsf{set}}\ \{\ \textsf{card}\ vs \mid \textsf{is_clique}\ vs\ g\ \}
\end{aligned}$$

What are we trusting now?

- e.g. HOL model of verified checkers and correspondence to real system
- HOL4 theorem prover, including logic, implementation, and execution environment [Slind and Norrish, 2008]

What are these right properties?

[Gocht, McCreesh, Myreen, Nordström, Oertel, and Tan, 2024; Ihalainen, Oertel, Tan, Berg, Järvisalo, Myreen, and Nordström, 2024]

$$
\begin{aligned}
&\mathsf{is\_clique}\ vs\ (v,e) \stackrel{\mathsf{def}}{=} \\
&\quad vs \subseteq \{\ 0,1,\ldots,v-1\ \} \land \\
&\quad \forall\, x\ y.\ x \in vs \land y \in vs \land x \neq y \Rightarrow \mathsf{is\_edge}\ e\ x\ y \\
&\mathsf{max\_clique\_size}\ g \stackrel{\mathsf{def}}{=} \mathsf{max}_{\mathsf{set}}\ \{\ \mathsf{card}\ vs \mid \mathsf{is\_clique}\ vs\ g\ \}
\end{aligned}
$$

What are we trusting now?

- e.g. HOL model of verified checkers and correspondence to real system
- HOL4 theorem prover, including logic, implementation, and execution environment [Slind and Norrish, 2008]

Proof logging in the Constraint Reasoning and Optimization Group

Earlier

- Fundamentals of redundancy notions in boolean decision problems (SAT) [Järvisalo, Heule, and Biere, 2012]

Currently

- Fundamentals of redundancy notions in boolean optimization (MaxSAT) [Berg and Järvisalo, 2019; Ihalainen, Berg, and Järvisalo, 2022]
- Certifying solvers and preprocessors [Ihalainen, Oertel, Tan, Berg, Järvisalo, Myreen, and Nordström, 2024; Berg, Bogaerts, Nordström, Oertel, and Vandesande, 2023]
- Multiobjective optimization [Jabs, Berg, Ihalainen, and Järvisalo, 2023]

Conclusion

Proof logging in automated reasoning:

- Guarantees correctness of results
- Supports development of increasingly complex reasoning into solvers.
- Provides audibility to third parties without access to the solver.

Open Challenges

- Practical scaling.
- Proof logging e.g. PSPACE-complete problems.
- Proving bounds on the proof systems used.

Conclusion

Proof logging in automated reasoning:

- Guarantees correctness of results
- Supports development of increasingly complex reasoning into solvers.
- Provides audibility to third parties without access to the solver.

Open Challenges

- Practical scaling.
- Proof logging e.g. PSPACE-complete problems.
- Proving bounds on the proof systems used.

I am hiring someone to work on these kinds of topics!

Bibliography I

Jeremias Berg and Matti Järvisalo. Unifying reasoning and core-guided search for maximum satisfiability. In Francesco Calimeri, Nicola Leone, and Marco Manna, editors, Logics in Artificial Intelligence - 16th European Conference, JELIA 2019, Rende, Italy, May 7-11, 2019, Proceedings, volume 11468 of Lecture Notes in Computer Science, pages 287–303. Springer, 2019. doi: 10.1007/978-3-030-19570-0_19. URL https://doi.org/10.1007/978-3-030-19570-0_19.

Jeremias Berg, Paul Saikko, and Matti Järvisalo. Subsumed label elimination for maximum satisfiability. In Gal A. Kaminka, Maria Fox, Paolo Bouquet, Eyke Hüllermeier, Virginia Dignum, Frank Dignum, and Frank van Harmelen, editors, ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016), volume 285 of Frontiers in Artificial Intelligence and Applications, pages 630–638. IOS Press, 2016. doi: 10.3233/978-1-61499-672-9-630. URL https://doi.org/10.3233/978-1-61499-672-9-630.

Jeremias Berg, Bart Bogaerts, Jakob Nordström, Andy Oertel, and Dieter Vandesande. Certified core-guided maxsat solving. In Brigitte Pientka and Cesare Tinelli, editors, Automated Deduction - CADE 29 - 29th International Conference on Automated Deduction, Rome, Italy, July 1-4, 2023, Proceedings, volume 14132 of Lecture Notes in Computer Science, pages 1–22. Springer, 2023. doi: 10.1007/978-3-031-38499-8_1. URL https://doi.org/10.1007/978-3-031-38499-8_1.

Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. Handbook of Satisfiability - Second Edition, volume 336 of Frontiers in Artificial Intelligence and Applications. IOS Press, 2021. ISBN 978-1-64368-160-3. doi: 10.3233/FAIA336. URL https://doi.org/10.3233/FAIA336.

Bart Bogaerts, Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. Certified symmetry and dominance breaking for combinatorial optimisation. In Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022, pages 3698–3707. AAAI Press, 2022. doi: 10.1609/AAAI.V36I4.20283. URL https://doi.org/10.1609/aaai.v36i4.20283.

Bibliography II

Jan Elffers and Jakob Nordström. A cardinal improvement to pseudo-boolean solving. In The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020, pages 1495–1503. AAAI Press, 2020. doi: 10.1609/AAAI.V34I02.5508. URL https://doi.org/10.1609/aaai.v34i02.5508.

Stephan Gocht, Ciaran McCreesh, Magnus O. Myreen, Jakob Nordström, Andy Oertel, and Yong Kiam Tan. End-to-end verification for subgraph solving. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada, pages 8038–8047. AAAI Press, 2024. doi: 10.1609/AAAI.V38I8.28642. URL https://doi.org/10.1609/aaai.v38i8.28642.

Marijn J. H. Heule. Proofs of unsatisfiability. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, Handbook of Satisfiability - Second Edition, volume 336 of Frontiers in Artificial Intelligence and Applications, pages 635–668. IOS Press, 2021. doi: 10.3233/FAIA200998. URL https://doi.org/10.3233/FAIA200998.

Marijn J. H. Heule, Benjamin Kiesl, and Armin Biere. Strong extension-free proof systems. J. Autom. Reason., 64(3):533–554, 2020. doi: 10.1007/S10817-019-09516-0. URL https://doi.org/10.1007/s10817-019-09516-0.

Hannes Ihalainen, Jeremias Berg, and Matti Järvisalo. Clause redundancy and preprocessing in maximum satisfiability. In Jasmin Blanchette, Laura Kovács, and Dirk Pattinson, editors, Automated Reasoning - 11th International Joint Conference, IJCAR 2022, Haifa, Israel, August 8-10, 2022, Proceedings, volume 13385 of Lecture Notes in Computer Science, pages 75–94. Springer, 2022. doi: 10.1007/978-3-031-10769-6_6. URL https://doi.org/10.1007/978-3-031-10769-6_6.

Bibliography III

Hannes Ihalainen, Andy Oertel, Yong Kiam Tan, Jeremias Berg, Matti Järvisalo, Magnus O. Myreen, and Jakob Nordström. Certified maxsat preprocessing. In Christoph Benzmüller, Marijn J. H. Heule, and Renate A. Schmidt, editors, Automated Reasoning - 12th International Joint Conference, IJCAR 2024, Nancy, France, July 3-6, 2024, Proceedings, Part I, volume 14739 of Lecture Notes in Computer Science, pages 396–418. Springer, 2024. doi: 10.1007/978-3-031-63498-7_24. URL https://doi.org/10.1007/978-3-031-63498-7_24.

Christoph Jabs, Jeremias Berg, Hannes Ihalainen, and Matti Järvisalo. Preprocessing in sat-based multi-objective combinatorial optimization. In Roland H. C. Yap, editor, 29th International Conference on Principles and Practice of Constraint Programming, CP 2023, August 27-31, 2023, Toronto, Canada, volume 280 of LIPIcs, pages 18:1–18:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi: 10.4230/LIPICS.CP.2023.18. URL https://doi.org/10.4230/LIPIcs.CP.2023.18.

Matti Järvisalo, Marijn Heule, and Armin Biere. Inprocessing rules. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings, volume 7364 of Lecture Notes in Computer Science, pages 355–370. Springer, 2012. doi: 10.1007/978-3-642-31365-3_28. URL https://doi.org/10.1007/978-3-642-31365-3_28.

Tuukka Korhonen, Jeremias Berg, Paul Saikko, and Matti Järvisalo. Maxpre: An extended maxsat preprocessor. In Serge Gaspers and Toby Walsh, editors, Theory and Applications of Satisfiability Testing - SAT 2017 - 20th International Conference, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings, volume 10491 of Lecture Notes in Computer Science, pages 449–456. Springer, 2017. doi: 10.1007/978-3-319-66263-3_28. URL https://doi.org/10.1007/978-3-319-66263-3_28.

Francesca Rossi, Peter van Beek, and Toby Walsh, editors. Handbook of Constraint Programming, volume 2 of Foundations of Artificial Intelligence. Elsevier, 2006. ISBN 978-0-444-52726-4. URL https://www.sciencedirect.com/science/bookseries/15746526/2.

Konrad Slind and Michael Norrish. A brief overview of HOL4. In TPHOLs, volume 5170 of LNCS, pages 28–32. Springer, 2008.

Bibliography IV

Jasper van Doornmalen, Leon Eifler, Ambros Gleixner, and Christopher Hojny. A proof system for certifying symmetry and optimality reasoning in integer programming, 2023.

Nathan Wetzler, Marijn Heule, and Warren A. Hunt Jr. Drat-trim: Efficient checking and trimming using expressive clausal proofs. In Carsten Sinz and Uwe Egly, editors, Theory and Applications of Satisfiability Testing - SAT 2014 - 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings, volume 8561 of Lecture Notes in Computer Science, pages 422–429. Springer, 2014. doi: 10.1007/978-3-319-09284-3_31. URL https://doi.org/10.1007/978-3-319-09284-3_31.