SMTCoq: Coq automation and its application to formal mathematics

Chantal Keller

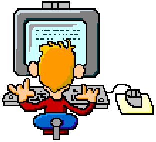
LRI, Université Paris-Saclay

Joint work with École polytechnique, Inria, Iowa University, Université Paris-Diderot

January, 8th 2020

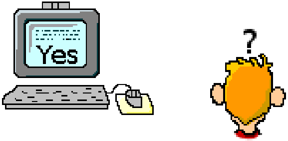
Motivation

INTERACTIVE THEOREM PROVER



$\longleftrightarrow$

AUTOMATIC THEOREM PROVER



SMT and Coq in a nutshell

	SAT/SMT solvers	Coq
Expressivity	First-order logic	**CIC**
Safety	Trust the whole software	**Kernel for proof checking**
Automation	**Automatic proof**	User-guided proof

↪ benefit from both worlds

Application 1: combinatorial mathematics

Example: Erdős Discrepancy Conjecture

For any infinite sequence $\langle x_1, x_2, \ldots \rangle$ of ± 1 integers and any integer C, there exist integers k and d such that

$$\left| \sum_{i=1}^{k} x_{i \times d} \right| > C$$

Proof for some C_0[1]:

Find l such that

$$\forall \langle x_1, x_2, \ldots, x_l \rangle, \forall k, \forall d, \left| \sum_{i=1}^{k} x_{i \times d} \right| \leqslant C_0$$

is unsatisfiable.

[1] This conjecture has been fully solved in 2015.

In Coq: computational reflection with efficient data-structures

```
Lemma erdos2_unsat :
   forall ρ, ~valid ρ erdos2.
Proof.
 apply (@checker_correct erdos2 ("erdos2.reso")).
 native_cast_no_check (refl_equal true).
Qed.
```

```
Lemma Erdos2 (x_ : nat → bool) :
  exists k, exists d,
    `|(\sum_(1 <= i < k) [x_ (i * d)])| > 2.
Proof.
  ...
  apply erdos2_unsat.
  ...
Qed.
```

(Credits: École polytechnique, Inria Sophia-Antipolis)

Application 2: tedious Coq proofs

```
Variable e : G, inv : G → G, op : G → G → G.

Hypothesis associative :
  forall a b c, op a (op b c) = op (op a b) c.
Hypothesis identity : forall a, (op e a = a).
Hypothesis inverse : forall a, (op (inv a) a = e).

Add_lemmas associative identity inverse.

Lemma identity' :
  forall a, (op a e = a).
Proof. smt. Qed.

Lemma inverse' :
  forall a, (op a (inv a) = e).
Proof. smt. Qed.

Lemma unique_identity e':
  (forall z, op e' z = z) → e' = e.
Proof. intros pe'; smt pe'. Qed.
```

(Credits: Iowa University, Université Paris-Diderot, Univ. Paris-Sud)

Example of CompCert

"Most of the development is conducted in first-order logic, suggesting the possibility of using automated theorem provers such as SMT solvers. Preliminary experiments with using SMT solvers to prove properties of the Compcert memory model indicate that many but not all of the lemmas can be proved automatically. While fully automated verification of a program like Compcert appears infeasible with today's technology, we expect that our interactive proof scripts would shrink significantly if Coq provided a modern SMT solver as one of its tactics."

A fragment of CompCert revisited with SMTCoq

```
Variable block, mem: Set.
Variable alloc_block: mem → Z → Z → block.
Variable alloc_mem: mem → Z → Z → mem.
Variable valid_block: mem → block → bool.

Hypothesis alloc_valid_block_1 m lo hi b :
  valid_block (alloc_mem m lo hi) b ⟶
    ((b =? (alloc_block m lo hi)) || valid_block m b).

Hypothesis alloc_valid_block_2 m lo hi b :
  ((b =? (alloc_block m lo hi)) || valid_block m b) ⟶
    valid_block (alloc_mem m lo hi) b.

Hypothesis alloc_not_valid_block m lo hi :
  negb (valid_block m (alloc_block m lo hi)).

Lemma alloc_valid_block_inv m lo hi b :
  valid_block m b → valid_block (alloc_mem m lo hi) b.
Proof. intro H. smt alloc_valid_block_2 H. Qed.

Lemma alloc_not_valid_block_2 m lo hi b' :
  valid_block m b' → b' =? (alloc_block m lo hi) = false.
Proof. intro H. smt alloc_not_valid_block H. Qed.
```

Challenges

- combinatorial mathematics: efficiency
- tedious Coq proofs: expressivity, encodings
- in addition: modularity with respect to solvers and theories

SMTCoq: in one system

- provers: ZChaff, glucose, veriT, CVC4
- "theories": equality, linear arithmetic, bit vectors, arrays, quantified hypotheses
- able to handle large combinatorial proofs
- Coq tactics that can combine theories

SMTCoq: skeptical approach

Certified ATP:

- prove the correctness of the code of the ATP

+ once and for all
+ completeness possible

- not flexible nor modular; freezes an implementation
- hard

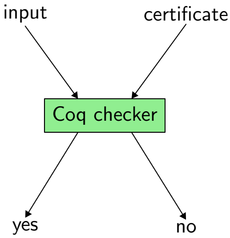
Certifying ATP:

- the ATP gives certificates that can be checked
- certificates to check each time (but efficient)
- no completeness (or at the meta-level)

+ very flexible and modular
+ easier (certified checker)

Outline

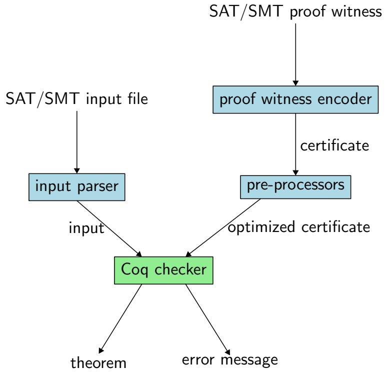
The heart: a certified checker for unsatisfiability



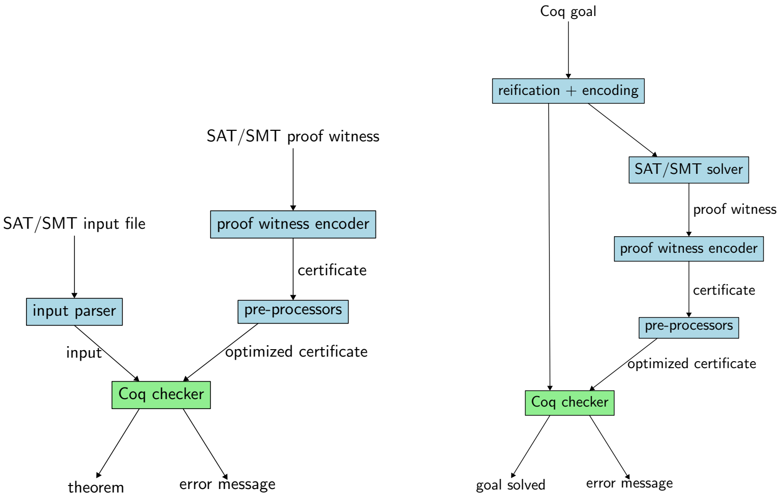
Certification:

- `checker : formula → certif → bool`
- `correctness` :
  $$\forall \phi\ \texttt{c}, \texttt{checker}\ \phi\ \texttt{c = true} \rightarrow \forall \rho, |\phi|_\rho = \texttt{false}$$
- $|\bullet|_\rho$: `formula → bool` is an interpretation function
- can be extracted to ML

The two applications



The two applications



Checker input and certificate formats

Input:

- a first-order formula ϕ in a combination of theories (SMT-LIB2)

Certificate:

- a proof of the unsatisfiability of ϕ in a combination of theories (in the large sense: modularity)

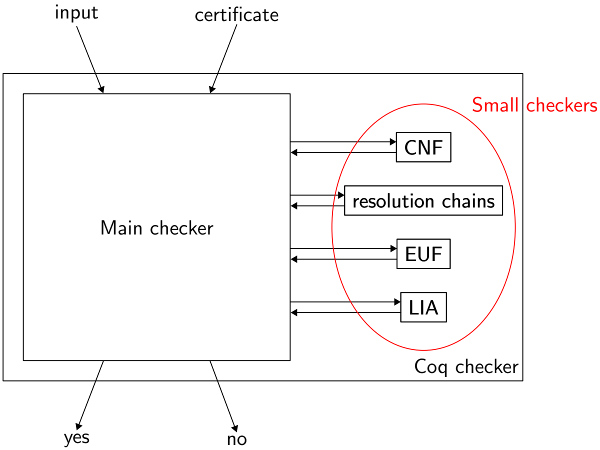
Example of a certificate

Unsatisfiability of (the conjunction of): $x \geq 7 \bigwedge y \leq -4 \qquad \neg x \geq 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{x \geq 7}\ \text{CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\ \text{LIA} \qquad \neg x \geq 2}{\neg x \geq 7}\ \text{Reso}}{\square}\ \text{Reso}$$

Outline

A modular checker



The small checkers and the main checker

A small checker:

- takes some clauses and a piece of certificate as arguments
- returns a clause that is implied

The main checker:

- maintains a set of clauses, initialized with the input
- sequentially shares out each certificate step between the corresponding small checker
- checks that the last obtained clause is the empty clause

The correctness of each small checkers implies the correctness of the whole checker

The main checker by example

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $\neg x \geq 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{x \geq 7}\ \text{CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\ \text{LIA} \qquad \neg x \geq 2}{\neg x \geq 7}\ \text{Reso}}{\square}\ \text{Reso}$$

The main checker by example

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $\neg x \geq 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{x \geq 7}\ \text{CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\ \text{LIA} \qquad \neg x \geq 2}{\neg x \geq 7}\ \text{Reso}}{\square}\ \text{Reso}$$

A set of clauses:

The main checker by example

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $\quad \neg x \geq 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{}\,\text{CNF} \qquad \dfrac{\dfrac{}{}\,\text{LIA} \qquad \neg x \geq 2}{}\,\text{Reso}}{}\,\text{Reso}$$

A set of clauses:

The main checker by example

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $\neg x \geq 2$

$$\cfrac{\cfrac{x \geq 7 \bigwedge y \leq -4}{}\ \text{CNF} \qquad \cfrac{\cfrac{}{}\ \text{LIA} \qquad \neg x \geq 2}{}\ \text{Reso}}{}\ \text{Reso}$$

A set of clauses:

$x \geq 7 \bigwedge y \leq -4$

$\neg x \geq 2$

The main checker by example

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $\neg x \geq 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{}\ \text{CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\ \text{LIA} \qquad \neg x \geq 2}{}\ \text{Reso}}{}\ \text{Reso}$$

A set of clauses:

$x \geq 7 \bigwedge y \leq -4$

$\neg x \geq 2$

The main checker by example

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $\neg x \geq 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{}\ \text{CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\ \text{LIA} \qquad \neg x \geq 2}{}\ \text{Reso}}{}\ \text{Reso}$$

A set of clauses:

$x \geq 7 \bigwedge y \leq -4$ $\neg x \geq 7 \vee x \geq 2$

$\neg x \geq 2$

The main checker by example

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $\qquad$ $\neg x \geq 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{}\ \text{CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\ \text{LIA} \qquad \neg x \geq 2}{\neg x \geq 7}\ \text{Reso}}{}\ \text{Reso}$$

A set of clauses:

$x \geq 7 \bigwedge y \leq -4$ $\qquad$ $\neg x \geq 7 \vee x \geq 2$

$\neg x \geq 2$

The main checker by example

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $\quad\quad \neg x \geq 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{}\,\text{CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\,\text{LIA} \qquad \neg x \geq 2}{\neg x \geq 7}\,\text{Reso}}{}\,\text{Reso}$$

A set of clauses:

$x \geq 7 \bigwedge y \leq -4$ $\quad \neg x \geq 7 \vee x \geq 2$

$\neg x \geq 2$ $\quad \neg x \geq 7$

The main checker by example

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $\neg x \geq 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{x \geq 7}\,\text{CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\,\text{LIA} \qquad \neg x \geq 2}{\neg x \geq 7}\,\text{Reso}}{}\,\text{Reso}$$

A set of clauses:

$x \geq 7 \bigwedge y \leq -4$ $\neg x \geq 7 \vee x \geq 2$

$\neg x \geq 2$ $\neg x \geq 7$

The main checker by example

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $\neg x \geq 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{x \geq 7}\,\text{CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\,\text{LIA} \qquad \neg x \geq 2}{\neg x \geq 7}\,\text{Reso}}{}\,\text{Reso}$$

A set of clauses:

$x \geq 7 \bigwedge y \leq -4$ $\neg x \geq 7 \vee x \geq 2$ $x \geq 7$

$\neg x \geq 2$ $\neg x \geq 7$

The main checker by example

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $\quad \neg x \geq 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{x \geq 7}\ \text{CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\ \text{LIA} \qquad \neg x \geq 2}{\neg x \geq 7}\ \text{Reso}}{\square}\ \text{Reso}$$

A set of clauses:

- $x \geq 7 \bigwedge y \leq -4$
- $\neg x \geq 7 \vee x \geq 2$
- $x \geq 7$
- $\neg x \geq 2$
- $\neg x \geq 7$

The main checker by example

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $\qquad \neg x \geq 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{x \geq 7}\,\text{CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\,\text{LIA} \qquad \neg x \geq 2}{\neg x \geq 7}\,\text{Reso}}{\square}\,\text{Reso}$$

A set of clauses:

$x \geq 7 \bigwedge y \leq -4$ $\qquad \neg x \geq 7 \vee x \geq 2$ $\qquad x \geq 7$

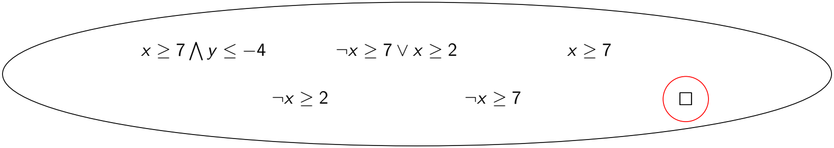
$\neg x \geq 2$ $\qquad \neg x \geq 7$ $\qquad \square$

The main checker by example

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $\qquad \neg x \geq 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{x \geq 7}\,\text{CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\,\text{LIA} \qquad \neg x \geq 2}{\neg x \geq 7}\,\text{Reso}}{\square}\,\text{Reso}$$

A set of clauses:



Example of optimization

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4 \qquad x < 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{x \geq 7}\,\text{CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\,\text{LIA} \qquad \neg x \geq 2}{\neg x \geq 7}\,\text{Reso}}{\square}\,\text{Reso}$$

Example of optimization

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $\quad$ $x < 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{x \geq 7}\text{ CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\text{ LIA} \qquad \neg x \geq 2}{\neg x \geq 7}\text{ Reso}}{\square}\text{ Reso}$$

3 clauses alive at the same time:

Example of optimization

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $x < 2$

$x \geq 7 \bigwedge y \leq -4$ CNF

LIA

$\neg x \geq 2$ Reso

Reso

3 clauses alive at the same time:

Example of optimization

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $\quad x < 2$

$$\cfrac{\cfrac{x \geq 7 \bigwedge y \leq -4}{} \text{CNF} \qquad \cfrac{\overline{}\ \text{LIA} \qquad \neg x \geq 2}{} \text{Reso}}{} \text{Reso}$$

3 clauses alive at the same time:

$x \geq 7 \bigwedge y \leq -4$	$\neg x \geq 2$	

Example of optimization

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $x < 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{}\ \text{CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\ \text{LIA} \qquad \neg x \geq 2}{}\ \text{Reso}}{}\ \text{Reso}$$

3 clauses alive at the same time:

$x \geq 7 \bigwedge y \leq -4$	$\neg x \geq 2$	

Example of optimization

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $\quad$ $x < 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{}\,\text{CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\,\text{LIA} \qquad \neg x \geq 2}{}\,\text{Reso}}{}\,\text{Reso}$$

3 clauses alive at the same time:

$x \geq 7 \bigwedge y \leq -4$	$\neg x \geq 2$	$\neg x \geq 7 \vee x \geq 2$

Example of optimization

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $\quad x < 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{}\ \text{CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\ \text{LIA} \qquad \neg x \geq 2}{\neg x \geq 7}\ \text{Reso}}{}\ \text{Reso}$$

3 clauses alive at the same time:

$x \geq 7 \bigwedge y \leq -4$	$\neg x \geq 2$	$\neg x \geq 7 \vee x \geq 2$

Example of optimization

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $x < 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{}\ \text{CNF} \quad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\ \text{LIA} \quad \neg x \geq 2}{\neg x \geq 7}\ \text{Reso}}{}\ \text{Reso}$$

3 clauses alive at the same time:

$x \geq 7 \bigwedge y \leq -4$	$\neg x \geq 2$	$\neg x \geq 7 \vee x \geq 2$
	↑	↑

Example of optimization

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $\quad x < 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{}\ \text{CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\ \text{LIA} \qquad \neg x \geq 2}{\neg x \geq 7}\ \text{Reso}}{}\ \text{Reso}$$

3 clauses alive at the same time:

$x \geq 7 \bigwedge y \leq -4$	$\neg x \geq 7$	$\neg x \geq 7 \vee x \geq 2$

Example of optimization

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $x < 2$

$$\cfrac{\cfrac{x \geq 7 \bigwedge y \leq -4}{x \geq 7}\text{CNF} \quad \cfrac{\cfrac{}{\neg x \geq 7 \vee x \geq 2}\text{LIA} \quad \neg x \geq 2}{\neg x \geq 7}\text{Reso}}{}\text{Reso}$$

3 clauses alive at the same time:

$x \geq 7 \bigwedge y \leq -4$	$\neg x \geq 7$	$\neg x \geq 7 \vee x \geq 2$

Example of optimization

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4 \qquad x < 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{x \geq 7}\,\text{CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\,\text{LIA} \qquad \neg x \geq 2}{\neg x \geq 7}\,\text{Reso}}{}\,\text{Reso}$$

3 clauses alive at the same time:

$x \geq 7 \bigwedge y \leq -4$	$\neg x \geq 7$	$\neg x \geq 7 \vee x \geq 2$

Example of optimization

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4 \qquad x < 2$

$$\frac{\dfrac{x \geq 7 \bigwedge y \leq -4}{x \geq 7}\,\text{CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\,\text{LIA} \qquad \neg x \geq 2}{\neg x \geq 7}\,\text{Reso}}{}\,\text{Reso}$$

3 clauses alive at the same time:

$x \geq 7$	$\neg x \geq 7$	$\neg x \geq 7 \vee x \geq 2$

Example of optimization

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $\quad$ $x < 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{x \geq 7}\,\text{CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\,\text{LIA} \qquad \neg x \geq 2}{\neg x \geq 7}\,\text{Reso}}{\square}\,\text{Reso}$$

3 clauses alive at the same time:

$x \geq 7$	$\neg x \geq 7$	$\neg x \geq 7 \vee x \geq 2$

Example of optimization

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $\quad x < 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{x \geq 7}\ \text{CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\ \text{LIA} \qquad \neg x \geq 2}{\neg x \geq 7}\ \text{Reso}}{\square}\ \text{Reso}$$

3 clauses alive at the same time:

$x \geq 7$	$\neg x \geq 7$	$\neg x \geq 7 \vee x \geq 2$
↑	↑	

Example of optimization

Unsatisfiability of: $x \geq 7 \bigwedge y \leq -4$ $x < 2$

$$\dfrac{\dfrac{x \geq 7 \bigwedge y \leq -4}{x \geq 7}\ \text{CNF} \qquad \dfrac{\dfrac{}{\neg x \geq 7 \vee x \geq 2}\ \text{LIA} \qquad \neg x \geq 2}{\neg x \geq 7}\ \text{Reso}}{\square}\ \text{Reso}$$

3 clauses alive at the same time:

$\square$	$\neg x \geq 7$	$\neg x \geq 7 \vee x \geq 2$

Pre-processing

Other optimizations:

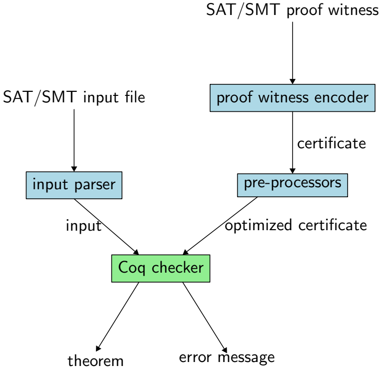
- certificates: cleaning, sharing
- formulas: hash-consing, flattening
- ...

Proof witness encoding (ZChaff, glucose, veriT, CVC4):

- fit into the format: encoding of nested proofs, DRUP
- recover information: arithmetic, simplifications, quantifiers, holes
- ...

$\hookrightarrow$ no need to certify: very easily extensible
$\hookrightarrow$ SMTCoq also safely combines SMT solvers

Application 1: combinatorial mathematics

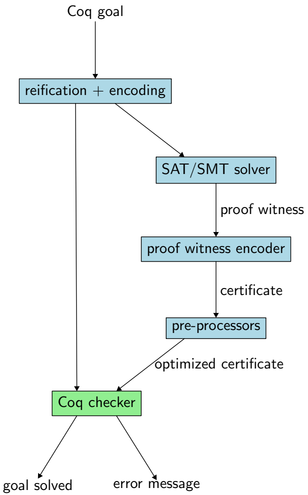


Efficiency:

- optimization
- native data-structures in Coq (integers, arrays)
- VM and native computation
- for very large witnesses: file reading on the fly

Outline

Application 2: Coq tactics



Expressivity:

- CIC → first-order logic
- take advantage of supported theories and quantifier instantiation
- cope with multiple definitions of the same mathematical objects
- deal with classical logic

Theories currently supported by SMTCoq

Current small checkers:

- extended resolution
- CNF computation
- equality
- linear integer arithmetic
- bit vectors
- arrays
- quantifiers
- silent simplifications

$\hookrightarrow$ modularity: they are independent (they only need to agree on formulae) $\Rightarrow$ easily extensible

Problem 1: multiple representations of similar objects

Integers:

- relative integers: Z, bigZ
- but also: nat, N, positive , bigN, . . .

Arrays:

- maps, arrays, functions, . . .

. . . and the same for most theories

. . . and the user may implement its own representation

A well-known problem

For integers:

- zify , ppsimpl: does not work in combination with other theories
- *transfer tactics*: requires human effort, works only for isomorphic types

↪ our own conversion tactics, applied before SMTCoq

Example

```
x, y : positive                    f : positive -> positive
============================================================
((x + 3) = y) -> ((3 < y) /\ ((f (x + 3)) <= (f y)))
```

↓

```
x', y' : Z         Hx' : 0 < x'      Hf'x' : 0 < f' (x' + 3)
f' : Z -> Z        Hy' : 0 < y'      Hf'y' : 0 < f' y'
=============================================================
((x' + 3) = y') -> ((3 < y') /\ (f' (x' + 3) <= f' y')))
```

Current approach

Ltac code:

1. add double conversions at the leaves
2. rewrite them bottom-up
3. rename to hide remaining conversions

Practical use:

- the user realizes a Coq module
- a functor automatically generates the tactic
- generic approach; implementation given for integers

Perspectives:

- provide implementations for all the theories
- robustness and mix with other SMTCoq features
- investigate a reflexive approach (MetaCoq)

Classical logic

Fact:
Classical logic is not needed in general, only in particular use cases that may appear in proofs

No axiom is added:

- automatic conversion of the quantifier-free part of the goal into a Boolean expression: the user may have to prove decidability of some predicates (or assume it)
- *shallow* treatment of quantifiers

$\hookrightarrow$ again, a conversion tactic from Prop to bool, applied before SMTCoq

Quantifiers

Expressivity of SMTCoq:

- goal: $\overline{\forall \overline{x}. P\ \overline{x}} \rightarrow \forall \overline{x}. G\ \overline{x}$
- context: $\overline{\forall \overline{x}. Q\ \overline{x}}$

($\overline{P}$, $\overline{Q}$, G quantifier-free)

In a nutshell:

SMTCoq can instantiate universally-quantified lemmas to prove the goal

Mixing deep and shallow embeddings in a reflexive tactic

Coq knows how to instantiate universal quantifiers!

+ let's use it
+ no binders in the type of formulae
+ intuitionistic
- a bit restrictive on the expressivity

Deep and shallow small checkers:

- conj_elim_l : formula *(* [A ∧ B] *)* → formula *(* [A] *)*
- forall_inst : (p:Prop) *(* forall x. P x *)* → (h:p) → formula *(* [P a] *)*

↪ an original way to write reflexive tactics

To obtain full first-order logic

Skolemization:

- need to carefully exhibit “classical” requirements

but in most use cases, lemma instantiation is sufficient

Perspectives of the tactics

Expressivity:

- inductive types and predicates (support from SMT solvers)
- dependent types (encoding)
- abduction
- ...

Robustness

Benchmarks:

- CompCert
- SSReflect and Mathematical Components

Outline

Play with it!

Combinatorial problems? Tedious Coq proofs?

`smtcoq.github.io`

User experience (and participation) welcome!