the boolean pythagorean triples problem in coq

luís cruz-filipe[1]
(joint work with joão marques-silva[2] and peter schneider-kamp[1])

[1]department of mathematics and computer science
university of southern denmark

[2]department of informatics
faculty of science, university of lisbon

logic and computation seminar, ist, lisbon
july 21st, 2017

outline

outline

original goal: verifying unsatisfiability

tacas'17

certifying (unsat) results from sat solvers

- enriched trace format
- verification procedure formalized in coq
- correct-by-construction extracted checker

original goal: verifying unsatisfiability

tacas'17

certifying (unsat) results from sat solvers

- enriched trace format
- verification procedure formalized in coq
- correct-by-construction extracted checker

evaluation

examples from the 2015 and 2016 sat competitions. . .

original goal: verifying unsatisfiability

tacas'17

certifying (unsat) results from sat solvers

- enriched trace format
- verification procedure formalized in coq
- correct-by-construction extracted checker

evaluation

examples from the 2015 and 2016 sat competitions...
...and "the large proof ever", because it's there

- unexpected success

the boolean pythagorean triples problem

a problem in ramsey theory

can the natural numbers be colored with two colors such that no pythagorean triple is monochromatic?

the boolean pythagorean triples problem

a problem in ramsey theory

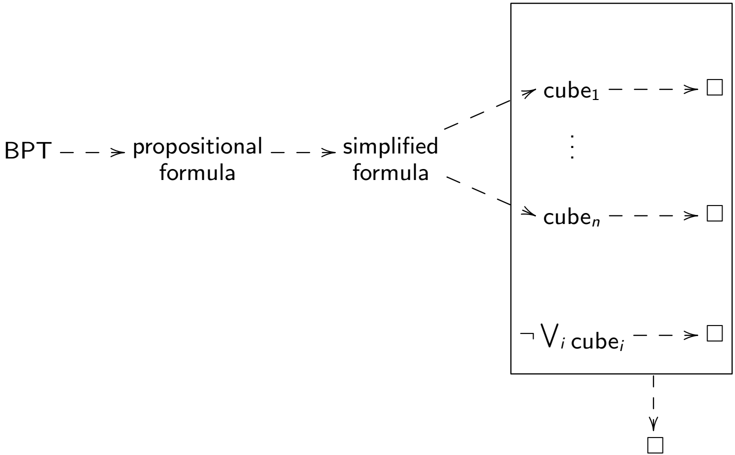
can the natural numbers be colored with two colors such that no pythagorean triple is monochromatic?

no

heule *et al.* showed that the finite restriction to $\{1, \ldots, 7825\}$ is already unsolvable

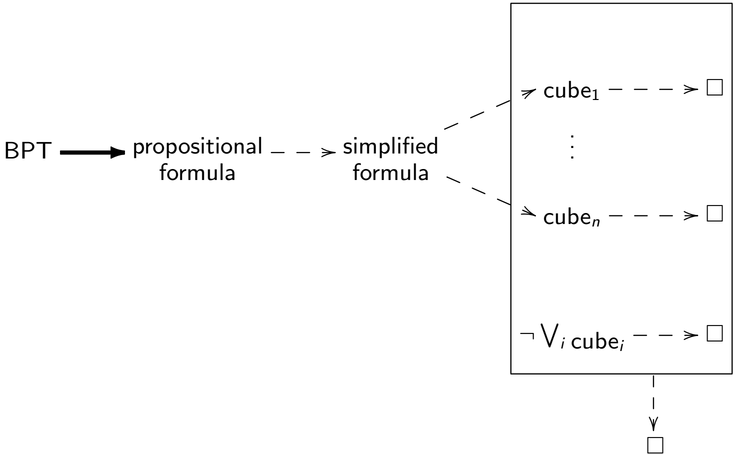
- encoding as a propositional formula
- simplification step
- divide-and-conquer strategy
- one million and one unsatisfiable formulas

proof strategy



outline

road map


BPT
propositional formula
simplified formula
cube$_1$
⋮
cube$_n$
$\neg \bigvee_i$ cube$_i$

the boolean pythagorean triples problem

definitions

- we use the coq type of (binary) positive numbers
- our "colors" are true and false

```
Definition coloring := positive -> bool.

Definition pythagorean (a b c:positive) := a*a + b*b = c*c.

Definition pythagorean_pos (C:coloring) := forall a b c,
  pythagorean a b c -> (C a <> C b) \/ (C a <> C c) \/ (C b <> C c).
```

a propositional encoding

```
Definition Pythagorean_formula (n:nat) := [...]
```

$$\bigwedge_{1 \leq a < b < c < n} (x_a \vee x_b \vee x_c) \wedge (\overline{x_a} \vee \overline{x_b} \vee \overline{x_c})$$

- (some) direct encoding in functional programming (we first build a list of pythagorean triples)
- n should be 7826, but it pays off to leave it uninstantiated

a propositional encoding

```
Definition Pythagorean_formula (n:nat) := [...]
```

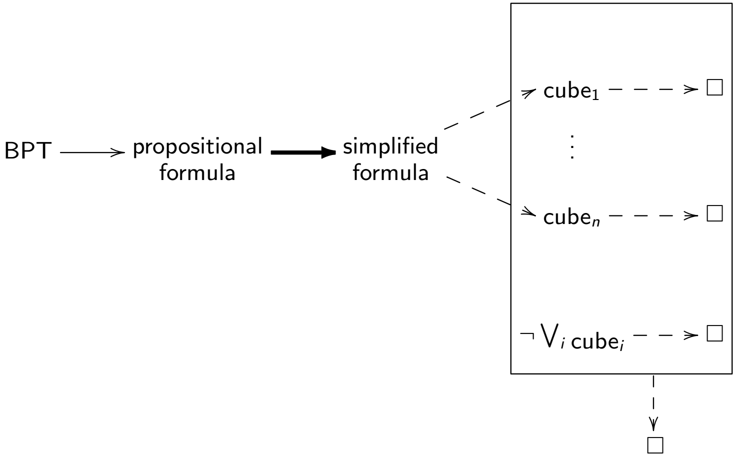
$$\bigwedge_{1\leq a<b<c<n} (x_a \vee x_b \vee x_c) \wedge (\overline{x_a} \vee \overline{x_b} \vee \overline{x_c})$$

- (some) direct encoding in functional programming (we first build a list of pythagorean triples)
- n should be 7826, but it pays off to leave it uninstantiated

```
Parameter TheN : nat.

Definition The_CNF := Pythagorean_formula TheN.

Theorem Pythagorean_Theorem : unsat The_CNF -> forall C, ~pythagorean_pos C.
```

- we can extract to ml and recompute the propositional formula

road map


BPT
propositional formula
simplified formula
$cube_1$
⋮
$cube_n$
$\neg \bigvee_i cube_i$

blocked clause elimination (i/ii)

in general

reduce the size of a cnf by eliminating clauses that do not change satisfiability

in this case

if k occurs in exactly one pythagorean triple, then that triple can be removed from the set

- any coloring that makes all remaining triples monochromatic can be trivially extended to k

blocked clause elimination (ii/ii)

```
Fixpoint simplify (t:triples) (l:list positive) := match l with
  | nil => t
  | p::l' => if (one_occurrence_dec p t) then simplify (remove_number p t) l'
             else simplify t l'
  end.

Definition simplified_Pythagorean_formula (n:nat) (l:list positive) := [...]

Parameter The_List : list positive.

Definition The_Simple_CNF := simplified_Pythagorean_formula TheN The_List.

Theorem simplification_ok : unsat The_CNF <-> unsat The_Simple_CNF.
```

- The_List is instantiated by a concrete list built from the trace of heule *et al.*'s proof

the symmetry break (i/ii)

idea

add additional constraints that preserve satisfiability but reduce the number of solutions
("without loss of generality...")

concretely

impose that 2520 is colored `true`

- nothing magical about 2520
- it just happen to be the number occurring most often

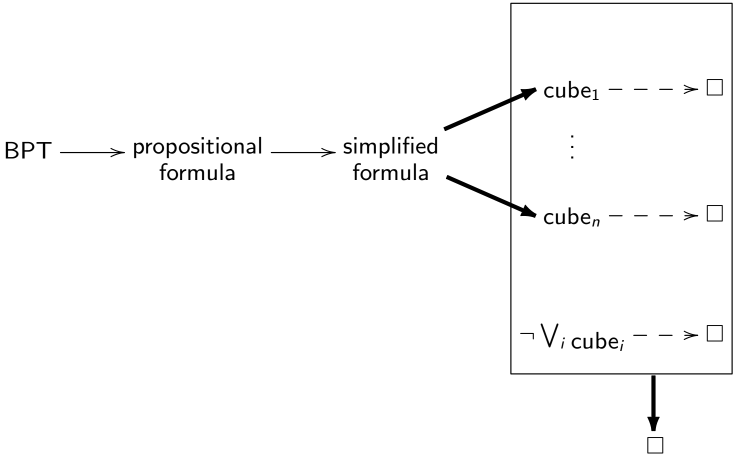
the symmetry break (ii/ii)

```
Lemma fix_one_color : forall C, pythagorean_pos C ->
  forall n b, exists C', pythagorean_pos C' /\ C' n = b.

Parameter TheBreak : positive.

Definition The_Asymmetric_CNF := [...]

Theorem symbreak_ok : unsat The_CNF <-> unsat The_Asymmetric_CNF.
```

- `The_Asymmetric_CNF` simply has the extra clause x_{2520}
- using program extraction we can compute the simplified propositional formula in approx. 35 minutes

outline

road map


BPT
propositional formula
simplified formula
cube_1
$\vdots$
cube_n
$\neg \bigvee_i \text{cube}_i$
$\square$

cube-and-conquer

methodology

find a set of partial valuations (the cubes) such that:

- the conjunction of the cnf with each cube is unsatisfiable
- the disjunction of the cubes is a tautology

a perfect balance

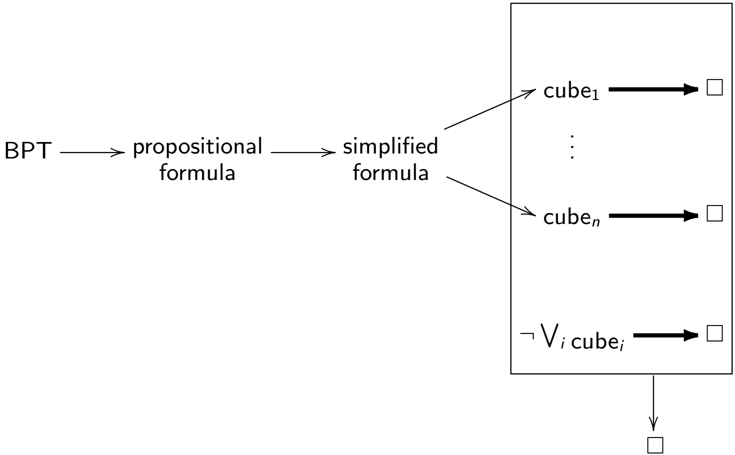
cubes are built using heuristics

- replace one big problem with many smaller ones
- need criteria to decide when to stop splitting
- not our problem!

cube-and-conquer, coq style

```
Definition Cube := list Literal.

Fixpoint Cubed_CNF (F:CNF) (C:Cube) : CNF := [...]

Fixpoint noCube (C:list Cube) : CNF := [...]

Lemma CubeAndConquer_lemma : forall Formula Cubes,
  (forall c, In c Cubes -> unsat (Cubed_CNF Formula c)) ->
  unsat (noCube Cubes) -> unsat Formula.
```

road map


BPT
propositional formula
simplified formula
cube$_1$
⋮
cube$_n$
$\neg \bigvee_i$ cube$_i$
□

verifying unsatisfiability

reverse unit propagation

we use an oracle containing lines of the form

$$\varphi, i_1, \ldots, i_k$$

- for each line, we check that φ follows from the current formula and add it
- we only allow reverse unit propagation, applying the clauses with indices $i_1, \ldots, i_k$ in sequence

verifying unsatisfiability

reverse unit propagation

we use an oracle containing lines of the form

$$\varphi, i_1, \ldots, i_k$$

- for each line, we check that φ follows from the current formula and add it
- we only allow reverse unit propagation, applying the clauses with indices $i_1, \ldots, i_k$ in sequence

- these are obtained from the 200 TB in heule *et al.*'s proof
- preprocessing yields the indices $i_1, \ldots, i_k$, which are the key to scalability
- the total size of the proof witnesses is nearly 400 TB

outline

conclusions

- formally verified unsolvability of the boolean pythagorean triples problem
- stronger claim for the mathematical result
- formal generation of the propositional encoding
- formal verification of the simplification process
- general certified framework for validation of proofs by cube-and-conquer
- general certified framework for unsatisfiability proofs (extended to a more expressive format, accepted at cade-26)

thank you!