The Boolean Pythagorean Triples Problem in Coq [*]

Luís Cruz-Filipe[1], Joao Marques-Silva[2], and Peter Schneider-Kamp[1]

[1] Department of Mathematics and Computer Science, University of Southern Denmark
{lcf,petersk}@imada.sdu.dk
[2] LaSIGE, Faculty of Science, University of Lisbon, Portugal
jpms@ciencias.ulisboa.pt

The Boolean Pythagorean Triples problem asks the following question: is it possible to partition the natural numbers into two sets such that no set contains a Pythagorean triple (three numbers a, b and c with $a^2+b^2=c^2$)? This question was answered in 2016, when Heule, Kullmann and Marek [4] showed that it is already impossible to partition the set $\{1,\ldots,7825\}$ into two sets such that none of them contains a Pythagorean triple. This proof was done by means of an encoding of this finite version of the problem into propositional logic (already used in [1]), which was then simplified and solved using the cube-and-conquer method [5].

The strategy of the proof is summarized in Figure 1. The propositional formula obtained by encoding the problem was first simplified using blocked clause elimination and symmetry breaking. Afterwards, the problem was divided into one million *cubes*: a set of partial assignments that cover the whole space of possible valuations. Then, it was shown that (1) the conjunction of the simplified formula with any cube is unsatisfiable, and (2) the negation of the disjunction of all the cubes is unsatisfiable. As a consequence, the simplified formula (and therefore also the original formula) is unsatisfiable.

This proof was formally verified using Coq, as described in [2, 3]. In this extended abstract, we summarize this process in a unified way.

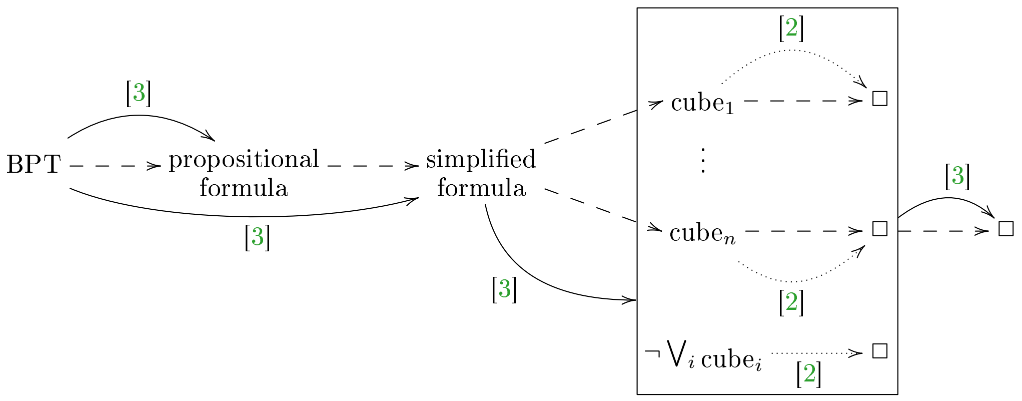

Figure 1: The original proof and the different verification steps. The dashed arrows denote the steps in the original proof [4]: a first propositional formula was generated by a C program, and subsequently simplified, divided and solved by SAT solvers. The dotted arrows denote proofs of unsatisfiability obtained by a SAT solver that were verified by a certified checker extracted from a Coq formalization [2]. The solid arrows denote the contributions of [3]: the generation, in Coq, of propositional formulas that are proved to represent the original mathematical problem, directly and after simplification; the formal specification of the simple reasoning behind cube-and-conquer; and the generation of the formulas that are given as input to cube-and-conquer.

[*]This work was partially supported by the Danish Council for Independent Research, Natural Sciences, grant DFF-1323-00247.

The first step was to formalize the Boolean Pythagorean Triples problem in Coq and relate it to the propositional encoding. This amounted to stating the mathematical problem in Coq and defining a family of propositional formulas (parametrized on a natural number n) directly corresponding to the formulas used in [4] as the starting point. We formally proved that unsatisfiability of any of these formulas implies that the given mathematical problem does not have a solution.

The second step was to formalize the simplification procedure. This is trivial, as the techniques applied in [4] only remove clauses, trivially preserving satisfiability. However, the authors of [4] make a stronger claim – namely, that their criterion for removing clauses also guarantees preservation of unsatisfiability. The mathematical argument, which we formalized, is as follows: let L be a list of triples and $(a, b, c) \in L$ be a triple containing a number that does not occur in any other triple in L. If there is a coloring C of the natural numbers such that no triple in $L \setminus \{(a, b, c)\}$ is monochromatic and e.g. a does not occur in any other triple in L, then we can extend C by changing the color of a, if necessary.

The next step was to formalize soundess of cube-and-conquer [5]. The idea behind this methodology is simple: instead of looking for a satisfying assignment for a particular formula, consider its conjunctions with different sets of literals (the cubes) such that every possible assignment satisfies one of the possible cubes. For example, if φ is a formula on two variables x and y, the cubes could be $\{x\}$, $\{\bar{x}, y\}$ and $\{\bar{x}, \bar{y}\}$, and instead of φ we consider the three formulas $\varphi \wedge x$, $\varphi \wedge \bar{x} \wedge y$ and $\varphi \wedge \bar{x} \wedge \bar{y}$. Furthermore, to ensure that every assignment satisfies one of the cubes, we need to check that the formula $\bar{x} \wedge (x \vee \bar{y}) \wedge (x \vee y)$ is unsatisfiable. We formalized this argument for the general case, given a formula and a list of cubes.

The last step was to check unsatisfiability of all the formulas generated by cube-and-conquer. This was done by developing a general-purpose verifier of unsatisfiability proofs based on reverse unit propagation [2]. This verifier, also formalized in Coq, checks that a given formula entails the empty clause by following a list of steps given as oracle. This list of steps is produced from a trace of an untrusted SAT solver, and is essentially a list of pairs (ψ, ℓ) where ψ is a clause to be added and ℓ is a list of indices of already known clauses that entail ψ by reverse unit propagation. (For efficiency, we also include deletion of clauses that are no longer relevant.)

Due to the huge size of the traces involved (over 200 TB), it is infeasible to perform the whole verification process inside Coq; thus, we use program extraction to obtain code that is correct by construction, and we rely on metalevel reasoning to chain the different steps in the process. However, we reduce this dependency to checking that the same arguments are provided to different functions. In principle, given enough resources, the whole verification could have been done inside Coq.

References

[1] J. Cooper and R. Overstreet. Coloring so that no pythagorean triple is monochromatic. *CoRR*, abs/1505.02222, 2015.

[2] L. Cruz-Filipe, J. Marques-Silva, and P. Schneider-Kamp. Efficient certified resolution proof checking. In *TACAS*, volume 10205 of *LNCS*, pages 118–135. Springer, 2017.

[3] L. Cruz-Filipe and P. Schneider-Kamp. Formally verifying the boolean pythagorean triples conjecture. In *LPAR 21*. EasyChair, accepted for publication.

[4] M. Heule, O. Kullmann, and V. W. Marek. Solving and verifying the boolean pythagorean triples problem via cube-and-conquer. In *SAT*, volume 9710 of *LNCS*, pages 228–245. Springer, 2016.

[5] M. Heule, O. Kullmann, S. Wieringa, and A. Biere. Cube and conquer: Guiding CDCL SAT solvers by lookaheads. In *HVC 2011*, volume 7261 of *LNCS*, pages 50–65. Springer, 2012.