▶ CARLOS CANTERO, BART BOGAERTS, AND DIETER VANDESANDE,
Proof logging the generalized totalizer encoding.
Department of Computer Science, KU Leuven, Celestijnenlaan 200A, Belgium.
E-mail: `carlos.cantero@kuleuven.be`.
Department of Computer Science, KU Leuven, Celestijnenlaan 200A, Belgium
and Artificial Intelligence Lab, Vrije Universiteit Brussel, Pleinlaan 9, Belgium.
E-mail: `bart.bogaerts@kuleuven.be`.
URL Address: `https://www.bartbogaerts.eu/`.
Department of Computer Science, KU Leuven, Celestijnenlaan 200A, Belgium
and Artificial Intelligence Lab, Vrije Universiteit Brussel, Pleinlaan 9, Belgium.
E-mail: `dieter.vandesande@kuleuven.be`.

An appealing practical application of proof theory is formal verification, a set of techniques that allow us to certify correctness properties of software, in particular allowing us to ensure that an algorithm will never fail. During the last few decades, various algorithms in which correctness is crucial have been formally verified, including the operating system of the fully automated Line 14 of the Paris Metro. Indeed, there are many critical systems for which plain testing is not enough, since we would like to be completely sure of their correctness.

We are interested in the discipline of combinatorial search and optimization, a class of techniques dedicated to finding, given a finite set of objects, an optimal one. Through numerous breakthroughs, these techniques are able to solve various NP-hard problems in reasonable time for practical inputs. But the rapid evolution of the state-of-the-art combinatorial optimization algorithms is a breeding ground for all kinds of bugs and safety concerns. Using formal verification techniques does not appear to be feasible in this setting: although they provide very strong guarantees, traditional formal verification techniques like model checking are slow and complicated processes that can hardly keep up with the fast progress in this field: a successful alternative is using *certifying algorithms*, which is also known as *proof logging* in this context.

Proof logging consists in making algorithms output a small proof of the correctness of a particular execution, which can later be checked by a much simpler proof checker, fit to be formally verified. Proof logging does not guarantee the general correctness of the algorithm, but it does ensure the correctness of every execution accepted by the proof checker. This offers both weaker and stronger guarantees with respect to traditional formal verification: weaker because they are less general, and stronger because they also certify every execution is free from isolated problems, like bit-flips due to cosmic rays. Proof logging also offers a great opportunity for auditability and provides useful debugging information when something does go wrong.

This is why we can observe an increased interest in proof logging techniques in the last two decades, as exemplified by the mandatory introduction of proof logging for SAT solvers in the main track of the SAT competition. Another discipline that can benefit from proof logging is the field of pseudo-Boolean solving, in particular the pseudo-Boolean-to-CNF encodings: the translation of pseudo-Boolean constraints into CNF formulas. A number of pseudo-Boolean solvers work by translating the input into CNF and then applying a SAT solver to the resulting clauses; at the moment, only the proof logging of the second part of this process has been extensively adopted.

There are many different PB-to-CNF encodings, one of which is the Generalized Totalizer Encoding (GTE). First introduced for cardinality constraints (pseudo-Boolean constraints in which every weight is equal to one) in [1], it was extended to general pseudo-Boolean constraints in [2]. The task of proof logging a PB-to-CNF encoding has two directions: *constraint to clauses*, where we prove that every model (satisfying assignment) of the original pseudo-Boolean constraint is also a model of the resulting encoding, and *clauses to constraint*, in which we prove the converse.[1] The direction *constraint to clauses* of the GTE was certified in [3], and it ensures we don't accidentally erase models during the encoding; this allows us, for example, to produce proofs of unsatisfiability, as exemplified by the proof logging of the iterative MaxSAT solver QMaxSAT [4]. A proof of the *clauses to constraint* direction would ensure we don't accidentally introduce new models during the encoding; an application of this property would be to certify techniques that need to preserve a certain correspondence of models between clauses and constraint, like pseudo-Boolean model counting.

In this work, we present a proof of the *clauses to constraint* direction, to the best of our knowledge for the first time. This proof can be used to ensure that projection onto the original variables induces a one-to-one correspondence between the models of the original pseudo-Boolean constraint and the models of the GTE encoding. To achieve that, we first show that the traditional definition of the GTE based in [2] is incomplete, in the sense that it introduces spurious models, and we present a simple extension to the definition that makes it complete. Following the lead of [3], we reason via a cutting planes derivation, in such a way that the proof can be readily transformed to a proof logging implementation in VeriPB style, where the cutting planes proof system [5] is an implicationally complete proof system that reasons with pseudo-Boolean constraints and VeriPB [6] is the leading proof checker for pseudo-Boolean proof logging.

[1] Olivier Bailleux and Yacine Boufkhad, *Efficient CNF Encoding of Boolean Cardinality Constraints*, ***Principles and Practice of Constraint Programming - CP 2003*** (Kinsale, Ireland), (Rossi, Francesca, editor), vol. 9, Springer Berlin Heidelberg, 2003, pp. 108–122.

[2] Saurabh Joshi and Ruben Martins and Vasco M. Manquinho, *Generalized Totalizer Encoding for Pseudo-Boolean Constraints*, ***Principles and Practice of Constraint Programming - CP 2015*** (Cork, Ireland), (Pesant, Gilles, editor), vol. 21, Springer International Publishing, 2015, pp. 200–209.

[3] Dieter Vandesande, *Towards Certified MaxSAT Solving: Certified MaxSAT solving with SAT oracles and encodings of pseudo-Boolean constraints*, ***Vrije Universiteit Brussel (VUB)***, https://researchportal.vub.be/nl/studentTheses/towards-certified-maxsat-solving.

[4] Dieter Vandesande, Wolf De Wulf and Bart Bogaerts, *QMaxSATpb: A Certified MaxSAT Solver*, ***Logic Programming and Nonmonotonic Reasoning - LPNMR 2022*** (Genova, Italy), (Gottlob, Georg and Inclezan, Daniela and Maratea, Marco, editors), vol. 16, Springer International Publishing, 2022, pp. 429–442.

[5] William J. Cook and Collette R. Coullard and György Turán, *On the complexity of cutting-plane proofs*, ***Discrete Applied Mathematics***, vol. 18 (1987), no. 1, pp. 25–38.

[6] Bart Bogaerts and Stephan Gocht and Ciaran McCreesh and Jakob Nordström, *Certified Dominance and Symmetry Breaking for Combinatorial Optimisation*, ***Journal of Artificial Intelligence Research***, vol. 77 (2023), pp. 1539–1589.

[1] More precisely, since in general the PB-to-CNF encodings introduce new variables, what we prove in the *clauses to constraint* direction is that every *projected model* of the encoding can be restricted to a model of the original constraint, so that in the end, if we prove both directions, we obtain a one-to-one correspondence of a certain kind of models rather than an equivalence.