Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence (IJCAI-24)
Sister Conferences Best Papers Track

Proof Logging for Smart Extensional Constraints (Extended Abstract)*

Matthew J. McIlree , Ciaran McCreesh
University of Glasgow, Glasgow, Scotland
m.mcilree.1@research.gla.ac.uk, ciaran.mccreesh@glasgow.ac.uk

Abstract

Proof logging provides an auditable way of guaranteeing that a solver has produced a correct answer using sound reasoning. This is standard practice for Boolean satisfiability solving, but for constraint programming, a challenge is that every propagator must be able to justify all inferences it performs. Here we demonstrate how to support proof logging for a wide range of previously uncertified global constraints. We do this by showing how to justify every inference that could be performed by the propagation algorithms for two families of generalised extensional constraint: "Smart Table" and "Regular Language Membership".

1 Overview

A proof log for a problem-solving algorithm provides a verifiable certificate that the result is correct, and also an auditable record of the steps taken to obtain that result. In the field of Boolean satisfiability (SAT), proof-logging has become an expected capability of modern solvers, with a series of proof formats including DRAT [Heule *et al.*, 2013a; Heule *et al.*, 2013b], LRAT [Cruz-Filipe *et al.*, 2017], and FRAT [Baek *et al.*, 2021] widely accepted for independent verification. A similar standard practice has not yet been adopted for Constraint Programming (CP) due to the difficulties of creating easily verifiable proofs for the more expressive formulations and reasoning present in this paradigm. However, recent work by Gocht et al. [Gocht *et al.*, 2022] has shown how the VeriPB proof system [Bogaerts *et al.*, 2023; Gocht, 2022] can be used to certify the reasoning carried out for several important expressive global constraints, offering a strong candidate for a complete, general CP proof logging method. In this setting, every propagator for a global constraint must be able to do two things: describe its semantics in a lower-level *pseudo-Boolean* format, and justify any reasoning it carries out using either *cutting planes* [Cook *et al.*, 1987] or *reverse unit propagation* (RUP) [Goldberg and Novikov, 2003] reasoning. Describing a constraint's semantics is a well-understood problem, but justifying propagation is not. Gocht et al. demonstrated proof logging for a range of global constraints and propagation strengths, including bounds-consistent integer linear inequalities and domain-consistent table constraints, but it remains an open question whether *every* global constraint propagation algorithm can be justified in this manner.

This work showed that the reasoning carried out by domain-consistent propagators for the `SmartTable` [Mairy *et al.*, 2015] and `Regular` [Pesant, 2004] constraints can similarly be proof logged efficiently inside the VeriPB proof system. As well as being useful in their own right, these two constraints provide the necessary building blocks for implementing many others, since they allow for efficient strong propagation for extensional constraints that cannot be expressed efficiently as a conventional table. For example, `SmartTable` can be used to implement the `Lex`, `AtMostOne` and `NotAllEqual` constraints [Mairy *et al.*, 2015], whilst `Regular` can be used to implement `Stretch`, `Geost` and `DiffN` [Lagerkvist and Pesant, 2008; Pesant, 2004]. Together, these two new constraints bring us a lot closer to fully auditable combinatorial solving, particularly in areas such as workforce scheduling where legal restrictions apply, and where algorithmic decisions can have a large effect upon people's livelihoods.

2 Implementation and Validation

We implemented both of these proof logging propagation algorithms inside the open-source *Glasgow Constraint Solver* [McCreesh and McIlree, 2023]. Our implementation added two new constraints `Regular` and `SmartTable`, with specific functionality to produce the required encodings and justifying propagators. They can therefore be used in conjunction with the rest of the constraint types already available in the solver.

We validated the implementations by first modelling some key examples. For `SmartTable`, these included the representation of lexicographic ordering problems, and problems where at most one variable takes a value, as given by Mairy et al. [Mairy *et al.*, 2015], as well as several illustrative examples. For `Regular`, we implemented both examples 1 and 2 from Pesant [Pesant, 2004].

*This paper was presented at the 29th International Conference on Principles and Practice of Constraint Programming, CP 2023 [McIlree and McCreesh, 2023]

We carried out further experimental validation by generating random (acyclic) smart tables, and random DFAs on sequences of up to five variables and solving the corresponding single-constraint problems. In all tests and examples, the solver produced an OPB model and proof files, and we checked these using the VeriPB proof checker. We also found that although proof logging incurs an obvious performance cost, the observed overheads were not unreasonable, giving a slowdown factor of between 2 and 10. As proofs are written currently written directly to disk, this can be very hardware dependent, and also dependent on how optimised the propagator implementation is. We decided to leave further engineering and optimisation of proof writing to future work.

During development some very subtle bugs in both propagator implementations that had eluded conventional testing were caught immediately by proof logging. For example, in an incremental version of the `Regular` propagator, an unsound inference was being made due to mixing up variable names, but in such a way that a situation where this unsound inference would actually lead to an incorrect solution was extremely rare (only in specially constructed instances, created once we were made aware of the bug due to proof logging). After correcting bugs such as these, all proofs were certified by VeriPB as being correct.

3 Conclusion

We have shown that we can efficiently justify all the reasoning that could possibly be carried out for two families of smart extensional constraints. An interesting observation is that justifying this reasoning required only the RUP rule, and this rule was only to provide hints of algorithmic steps that were already being carried out by the propagators. In effect, we are logging a sequence of "lookahead to see immediate contradiction" steps. We did not require any explicit cutting planes derivations, and although we relied upon pseudo-Boolean constraints to make it simple to express reifications and negations, in principle everything we did should also be possible in a weaker proof encoding and proof system such as CNF and DRAT. The only caveat is that we *do* rely upon strong propagation properties for encoded integer variables, which would limit approaches based upon Boolean satisfiability to integer variables with very small domains. This is in contrast to constraints like `AllDifferent` and `Linear`, which cannot be logged efficiently in resolution-based approaches [Elffers *et al.*, 2020; Gocht *et al.*, 2022].

We expect that other global constraints, even those with complex propagation algorithms will be similarly feasible to proof log using this technique. Richer smart tables, such as those with offset or ternary restrictions [Boussemart *et al.*, 2016], also fit into our framework, although the justification of the filtering inferences for each restriction may require additional cutting planes steps. Furthermore, the propagation of "Multi-Valued Decision Diagram"-based constraints [Hoda *et al.*, 2010] can be viewed as a generalisation of the techniques used for the regular language membership constraint, and so it seems very plausible that a similar proof logging methodology as demonstrated in this paper could work here. This bodes well for being able to provide auditable solving for most global constraints that might occur in a modern solver.

Acknowledgments

The authors would like to thank Jakob Nordström for several helpful discussions regarding pseudo-Boolean encodings and unit propagation. Ciaran McCreesh was supported by a Royal Academy of Engineering research fellowship. Part of this work was done while the authors were participating in a programme at the Simons Institute for the Theory of Computing.

References

[Baek *et al.*, 2021] Seulkee Baek, Mario Carneiro, and Marijn J. H. Heule. A flexible proof format for SAT solver-elaborator communication. In *Tools and Algorithms for the Construction and Analysis of Systems: 27th International Conference, TACAS 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 – April 1, 2021, Proceedings, Part I*, page 59–75, Berlin, Heidelberg, 2021. Springer-Verlag.

[Bogaerts *et al.*, 2023] Bart Bogaerts, Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. Certified dominance and symmetry breaking for combinatorial optimisation. *J. Artif. Intell. Res.*, 77:1539–1589, 2023.

[Boussemart *et al.*, 2016] Frédéric Boussemart, Christophe Lecoutre, and Cédric Piette. XCSP3: an integrated format for benchmarking combinatorial constrained problems. *CoRR*, abs/1611.03398, 2016.

[Cook *et al.*, 1987] William J. Cook, Collette R. Coullard, and György Turán. On the complexity of cutting-plane proofs. *Discret. Appl. Math.*, 18(1):25–38, 1987.

[Cruz-Filipe *et al.*, 2017] Luís Cruz-Filipe, Marijn J. H. Heule, Warren A. Hunt, Matt Kaufmann, and Peter Schneider-Kamp. Efficient certified RAT verification. In Leonardo de Moura, editor, *Automated Deduction – CADE 26*, pages 220–236, Cham, 2017. Springer International Publishing.

[Elffers *et al.*, 2020] Jan Elffers, Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. Justifying All Differences Using Pseudo-Boolean Reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(02):1486–1494, April 2020.

[Gocht *et al.*, 2022] Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. An Auditable Constraint Programming Solver. In Christine Solnon, editor, *28th International Conference on Principles and Practice of Constraint Programming (CP 2022)*, volume 235 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:18, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[Gocht, 2022] Stephan Gocht. *Certifying Correctness for Combinatorial Algorithms by Using Pseudo-Boolean Reasoning*. PhD thesis, Lund University, Sweden, 2022.

[Goldberg and Novikov, 2003] Evguenii I. Goldberg and Yakov Novikov. Verification of proofs of unsatisfiability for CNF formulas. In *2003 Design, Automation and Test in Europe Conference and Exposition (DATE 2003), 3-7 March 2003, Munich, Germany*, pages 10886–10891. IEEE Computer Society, 2003.

[Heule *et al.*, 2013a] Marijn J. H. Heule, Warren A. Hunt, and Nathan Wetzler. Verifying refutations with extended resolution. In Maria Paola Bonacina, editor, *Automated Deduction – CADE-24*, pages 345–359, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[Heule *et al.*, 2013b] Marijn J.H. Heule, Warren A. Hunt, and Nathan Wetzler. Trimming while checking clausal proofs. In *2013 Formal Methods in Computer-Aided Design*, pages 181–188, 2013.

[Hoda *et al.*, 2010] Samid Hoda, Willem Jan van Hoeve, and John N. Hooker. A systematic approach to MDD-based constraint programming. In David Cohen, editor, *Principles and Practice of Constraint Programming - CP 2010 - 16th International Conference, CP 2010, St. Andrews, Scotland, UK, September 6-10, 2010. Proceedings*, volume 6308 of *Lecture Notes in Computer Science*, pages 266–280. Springer, 2010.

[Lagerkvist and Pesant, 2008] Mikael Z Lagerkvist and Gilles Pesant. Modeling irregular shape placement problems with regular constraints. In *First workshop on bin packing and placement constraints BPPC'08*, 2008.

[Mairy *et al.*, 2015] Jean-Baptiste Mairy, Yves Deville, and Christophe Lecoutre. The Smart Table Constraint. In Laurent Michel, editor, *12th International Conference on Integration of AI and OR Techniques in Constraint Programming (CPAIOR 2015)*, Lecture Notes in Computer Science, pages 271–287, Cham, 2015. Springer International Publishing.

[McCreesh and McIlree, 2023] Ciaran McCreesh and Matthew McIlree. The Glasgow Constraint Solver. GitHub repository, 2023.

[McIlree and McCreesh, 2023] Matthew J. McIlree and Ciaran McCreesh. Proof logging for smart extensional constraints. 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 26:1–26:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[Pesant, 2004] Gilles Pesant. A Regular Language Membership Constraint for Finite Sequences of Variables. In Mark Wallace, editor, *10th International Conference on Principles and Practice of Constraint Programming (CP 2004)*, Lecture Notes in Computer Science, pages 482–495, Berlin, Heidelberg, 2004. Springer.