# End-to-End Certified Graph Colouring (Extended Abstract)[1]

**Simon Dold**
University of Basel, Switzerland

**George Katsirelos**
MIA Paris, INRAE, Université Paris-Saclay, France
Université de Toulouse, ANITI, France

**Wietze Koops**[2]
Lund University, Sweden
University of Copenhagen, Denmark

**Magnus O. Myreen**
Chalmers University of Technology, Gothenburg, Sweden
University of Gothenburg, Sweden

**Jakob Nordström**
University of Copenhagen, Denmark
Lund University, Sweden

**Andy Oertel**
Lund University, Sweden
University of Copenhagen, Denmark

**Yong Kiam Tan**
Nanyang Technological University, Singapore
Institute for Infocomm Research ($I^2R$), A*STAR, Singapore

The *graph colouring* problem is to assign colours to the vertices of a graph so that for every edge the two endpoints have different colours. Determining the minimum number of colours needed—the so-called *chromatic number*—is one of the most well-studied NP-hard problems in theoretical computer science [1, 14, 25], and is also important for many practical applications, including compiler optimization [4], numerical analysis [7], and scheduling [16]. Impressive progress in applied research [3, 6, 11] has led to solvers that can find optimal colourings even for very large graphs in spite of the theoretical hardness results. However, this leap in performance has come at the cost of increasing solver complexity, which makes it challenging to avoid errors in algorithm design and implementation. Due to the use of reductions and reformulations, it can even be tricky to recover solutions to the original problem. This means that graph colouring is afflicted by the same concerns as in other areas of combinatorial optimization, where even mature solvers sometimes erroneously claim optimality or return "solutions" which fail to satisfy all constraints [8, 22].

The Boolean satisfiability (SAT) community has pioneered the use of *certifying solvers* to address this problem. Certifying solvers use *proof logging* to output a machine-verifiable proof that the answer the solver produced is correct. In the now de facto standard DRAT [23] format, such a proof essentially consists of just the clauses that the solver has learned. As a result, the overhead of proof generation is generally at most 10% of the solving time, while proof checking can be done within a factor 10 of the solving time. Proof checkers also come with a formally verified backend, which means that correctness is certified by the strongest guarantees offered by formal methods [21].

The most successful approach to port these successes to more expressive paradigms is *pseudo-Boolean (PB) proof logging*, which uses 0–1 linear constraints, and has been applied in SAT-based optimization (MaxSAT) [2, 13], subgraph solving [9, 10], and constraint programming [17, 18], among others. However, the overhead for proof generation and checking has mostly been significantly larger for paradigms beyond SAT.

[1] This extended abstract presented at the CP/SAT Doctoral Program summarizes and as such builds heavily on the corresponding CP paper [5]. We refer to the CP paper for declaration of funding, acknowledgements, and references to supplementary material.
[2] Student submitting to SAT/CP Doctoral Program



Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

In this work, we present for the first time a state-of-the-art graph colouring solver with proof logging, namely a version of ZykovColor [3] emitting VeriPB proofs.

ZykovColor is based on the Zykov recurrence [26] combined with SAT solving, using *reified constraints* to define new variables. To compute chromatic number lower bounds the solver uses cliques but also detects *Mycielski graphs* [20], for which our proof logging uses and extends work by Yolcu et al. [24]. The only technique for which we currently cannot provide proof logging is the fractional chromatic number bound computed via LP duality and column generation [12, 19], since VeriPB cannot reason about rational values, but this feature has only a small impact on solver performance [3]. It has been shown previously that proof logging can serve as a powerful debugging tool even for mature solvers [2, 15, 22], and our work on making ZykovColor certifying provides yet another example of this in that it helped fix a bug where a colouring with too many colours could be returned.

VeriPB proofs reason in terms of a 0–1 integer linear encoding of the graph colouring problem instance. Though this encoding is quite straightforward, our proof checking workflow includes formal verification of the translation by a CakePB frontend. Since CakePB also has a formally verified backend for checking VeriPB proofs, we achieve end-to-end formally certified results (similar to prior work [9]). An additional optimization is that an initial upper bound on the chromatic number can be used to shrink encoding size substantially.

Our experimental evaluation shows excellent performance overall, with average proof logging overhead of under 14% and average checking time of 1.4× solving time, but there are also some problematic instances. For graphs with large cliques, the proofs need to rederive for every colour that at most one vertex in the clique can take this colour (since all variables are Boolean), while the solver automatically knows that this holds for any colour. This causes an extra linear overhead for proof generation, similar to phenomena that have been observed in proof logging for constraint programming [17, 18]. We also incur slowdowns for Mycielski graphs of high order, since the pseudo-Boolean proofs for their chromatic number formalize a rather subtle recursive argument that has to be explicitly written out and checked, whereas the solver can just use that the theorem can be trusted to be true.

We should emphasize that our performance statistics were achieved only after several rounds of careful optimization of the proof logging. For instance, it was crucial to generate constraints as lazily as possible, and previous proofs for Mycielski graphs [24] were made more efficient. For cliques with large intersections, it was important to recycle subderivations for these intersections. This shows that while pseudo-Boolean reasoning makes efficient proof logging possible even for advanced combinatorial algorithms, getting the low-level details right to achieve good performance in practice is arguably still a bit too complicated for comfort.

The proof checking performance also hinges on a large number of low-level optimizations, many of which do not seem to be related to graph colouring. Since writing an efficient proof checker, including a formally verified backend, is immensely challenging, this provides an additional argument in favour of a simple, unified proof logging format such as VeriPB, which can be combined with different formally verified frontends for a wide range of problems.

## References


1 Paul Beame, Joseph C. Culberson, David G. Mitchell, and Cristopher Moore. The resolution complexity of random graph $k$-colorability. *Disc. Appl. Math.*, 153(1-3):25–47, 2005.

2 Jeremias Berg, Bart Bogaerts, Jakob Nordström, Andy Oertel, and Dieter Vandesande. Certified core-guided MaxSAT solving. In *CADE-29*, volume 14132 of *LNCS*, pages 1–22, 2023.

3 Timo Brand, Daniel Faber, Stephan Held, and Petra Mutzel. A customized SAT-based solver for graph coloring. In *ALENEX*, pages 142–155. SIAM, 2026.

4 Gregory J. Chaitin. Register allocation & spilling via graph coloring. *SIGPLAN Not.*, 17(6):98–101, June 1982.

5 Simon Dold, George Katsirelos, Wietze Koops, Magnus O. Myreen, Jakob Nordström, Andy Oertel, and Yong Kiam Tan. End-to-end certified graph colouring. In *CP '26*, 2026. To appear.

6 Daniel Faber, Adalat Jabrayilov, and Petra Mutzel. SAT Encoding of Partial Ordering Models for Graph Coloring Problems. In *SAT '24*, volume 305 of *LIPIcs*, pages 12:1–12:20, 2024.

7 Assefaw Hadish Gebremedhin, Fredrik Manne, and Alex Pothen. What color is your Jacobian? graph coloring for computing derivatives. *SIAM Review*, 47(4):629–705, 2005.

8 Xavier Gillard, Pierre Schaus, and Yves Deville. SolverCheck: Declarative testing of constraints. In *CP '19*, volume 11802 of *LNCS*, pages 565–582, 2019.

9 Stephan Gocht, Ciaran McCreesh, Magnus O. Myreen, Jakob Nordström, Andy Oertel, and Yong Kiam Tan. End-to-end verification for subgraph solving. In *AAAI '24*, pages 8038–8047, 2024.

10 Stephan Gocht, Ciaran McCreesh, and Jakob Nordström. Subgraph isomorphism meets cutting planes: Solving with certified solutions. In *IJCAI '20*, pages 1134–1140, 2020.

11 Emmanuel Hebrard and George Katsirelos. Constraint and satisfiability reasoning for graph coloring. *Journal of Artificial Intelligence Research*, 69:33–65, 2020.

12 Stephan Held, William Cook, and Edward C. Sewell. Maximum-weight stable sets and safe lower bounds for graph coloring. *Mathematical Programming Computation*, 4(4):363–381, 2012.

13 Hannes Ihalainen, Andy Oertel, Yong Kiam Tan, Jeremias Berg, Matti Järvisalo, Magnus O. Myreen, and Jakob Nordström. Certified MaxSAT preprocessing. In *IJCAR '24*, volume 14739 of *LNCS*, pages 396–418, 2024.

14 Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Springer, 1972.

15 Wietze Koops, Daniel Le Berre, Magnus O. Myreen, Jakob Nordström, Andy Oertel, Yong Kiam Tan, and Marc Vinyals. Practically feasible proof logging for pseudo-Boolean optimization. In *CP '25*, volume 340 of *LIPIcs*, pages 21:1–21:27, 2025.

16 Vahid Lotfi and Sanjiv Sarin. A graph coloring algorithm for large scale scheduling problems. *Computers & Operations Research*, 13(1):27–32, 1986.

17 Matthew McIlree and Ciaran McCreesh. Proof logging for smart extensional constraints. In *CP '23*, volume 280 of *LIPIcs*, pages 26:1–26:17, 2023.

18 Matthew McIlree, Ciaran McCreesh, and Jakob Nordström. Proof logging for the circuit constraint. In *CPAIOR '24*, volume 14743 of *LNCS*, pages 38–55, 2024.

19 Anuj Mehrotra and Michael A. Trick. A column generation approach for graph coloring. *INFORMS Journal on Computing*, 8(4):344–354, 1996. doi:10.1287/ijoc.8.4.344.

20 Jan Mycielski. Sur le coloriage des graphs. *Colloquium Mathematicae*, 3(2):161–162, 1955.

21 Yong Kiam Tan, Marijn J. H. Heule, and Magnus O. Myreen. cake_lpr: Verified propagation redundancy checking in CakeML. In *TACAS '21*, volume 12652 of *LNCS*, pages 223–241. Springer, 2021.

22 Cesare Tinelli. Scalable proof production and checking in SMT. In *SAT '24*, volume 305 of *LIPIcs*, pages 2:1–2:2, August 2024.

23 Nathan Wetzler, Marijn J. H. Heule, and Warren A. Hunt Jr. DRAT-trim: Efficient checking and trimming using expressive clausal proofs. In *SAT '14*, volume 8561 of *LNCS*, pages 422–429. Springer, 2014.

24 Emre Yolcu, Xinyu Wu, and Marijn J. H. Heule. Mycielski graphs and PR proofs. In *SAT '20*, pages 201–217, 2020.

25 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(6):103–128, August 2007.

26 Alexander Aleksandrovich Zykov. On some properties of linear complexes. *Matematicheskii Sbornik*, 66(2):163–188, 1949.