2025 Continuity Computability Constructivity
From Logic to Algorithms

Verifying Z3 SAT Proofs with the Interactive Theorems Prover Coq/Rocq and Agda

Harry Bryant[1]*, Andrew Lawrence[2],
Monika Seisenberger[1]†, and
Anton Setzer[1]‡

[1] Swansea University, Dept. of Computer Science, Swansea, Wales, UK
{harry.bryant,m.seisenberger,a.g.setzer}@swansea.ac.uk
[2] Siemens Mobility, Chippenham, England, UK andrew.lawrence@siemens.com

Ensuring the correctness of safety-critical systems, such as railway control systems, remains paramount. To achieve this, machine-assisted theorem proving is increasingly adopted in the railway domain. Tools like Z3 [17] are used to formally verify that these systems meet stringent safety standards. For example, our group has applied such techniques in the verification of geographic scheme data and ladder logic interlockings [2, 11]. These approaches help identify design flaws early, reducing the risk and cost associated with later-stage testing and certification. However, any of these solvers may have flaws or implement optimisations that produce incorrect results. To increase trust, proof checking offers an independent check of the Z3 output. We are developing a verified SAT proof checker for Z3 using its new Reverse Unit Propagation (RUP) format. We have also incorporated a checking procedure for the Tseitin Transformation to achieve propositional formulae in conjunctive normal form (CNF) [26]. The new RUP proof format was introduced to the Z3 theorem prover in September 2022, replacing resolution [7]. Proof checking for other proof formats for SAT and SMT solving has been performed in, e.g., [8, 10, 9, 12, 41, 15, 43, 21, 13, 27, 41, 42, 36, 25, 16, 18, 1, 20, 30, 3, 5, 4, 19, 31, 33, 35, 6].

The notion of a RUP proof was introduced by van Gelder [23, 22] in 2008. It addresses the issue that resolution proofs can be too lengthy to store feasibly while still allowing efficient checking. The underlying concept is proof verification by Goldberg and Novikov [24], where unit propagation checks unsatisfiability without storing full resolution proofs. Van Gelder refined this into RUP, requiring each derived clause to cause a contradiction when added, making proofs more compact and efficient. RUP takes logical statements written in CNF, where each clause is a disjunction of literals $\{x_1, \ldots, x_n\}$. Negation of a literal x_i simply switches from x_i to $\neg x_i$, or vice versa. A formula is a conjunction of clauses. Z3 deals with formulae not in CNF by translating them using the Tseitin transformation [40, 34, 29].

A RUP inference of a clause $C = \{x_1, \ldots, x_n\}$ from assumptions (clauses) Γ is correct, if from $\Gamma' := \Gamma, \{\neg x_1\}, \ldots, \{\neg x_n\}$ we can derive the empty clause $\{\}$ using only unit-clause propagation. A RUP proof from an initial clause set Γ_0 is a sequence of clauses C_i, for $i \geqslant 1$, such that for all i C_i is a RUP inference from Γ_{i-1}, where $\Gamma_j = \Gamma_{j-1} \cup \{C_j\}$, for $j \geqslant 1$. If some C_j is the empty clause $\{\}$, the sequence is a RUP refutation [23]. Checking a RUP inference is done as follows: Divide Γ' into non-unit clauses Γ_{nunit} (clauses of length $\geqslant 2$) and unit clauses Γ_{unit} (clauses of length 1). If an empty clause is found, then we have derived falsity, and Γ was already unsatisfiable. While $\Gamma_{\text{unit}} \neq \emptyset$, we repeatedly select a unit clause x from Γ_{unit}, remove it, and apply unit resolution with all clauses in $\Gamma_{\text{unit}} \cup \Gamma_{\text{nunit}}$. If c contains x, then it is implied by $\{x\}$ and is therefore removed. Otherwise, if c contains $\neg x$, then a unit resolution of c with

*https://github.com/HarryBryant99, ORCID: 0009-0008-9926-8678
†https://www.swansea.ac.uk/staff/m.seisenberger/, ORCID: 0000-0002-2226-386X
‡https://csetzer.github.io/, https://www.swansea.ac.uk/staff/a.g.setzer/, ORCID: 0000-0001-5322-6060

$\{x\}$ derives $c' := c \backslash \{\neg x\}$. We replace c by c'; if it has length $\geqslant 2$, it will be in Γ_{nunit}, and if it has length 1, in Γ_{unit}. If it has length 0, then we have derived $\{\}$, so the RUP inference is verified, and we exit the loop. If c does not contain x or $\neg x$, it is kept. Once we have applied unit resolution with $\{x\}$ to all the clauses in $\Gamma_{\text{unit}} \cup \Gamma_{\text{nunit}}$, we repeat the process. After each step, the literal x and its negation do not occur anymore in $\Gamma_{\text{unit}} \cup \Gamma_{\text{nunit}}$, and no new literals have been created, so eventually the loop terminates because Γ_{unit} is empty. If we have not derived $\{\}$ by then, then $\{\}$ is not derivable by unit clause propagation, thus the verification of the RUP inference fails. At each step, all formulae in $\Gamma_{\text{unit}} \cup \Gamma_{\text{nunit}}$ are derivable from Γ using unit clause resolution; therefore, they are entailed. If the procedure succeeds, then Γ' entails falsity and is therefore unsatisfiable. Thus, by classical logic (we have tertium non datur for the Boolean variables) it follows that Γ entails $\{x_1, \ldots, x_n\}$.

We have formalised our Z3 SAT proof-checker in Rocq [37], previously known as Coq [14], and developed a procedure to validate Z3-generated proofs using the Reverse Unit Propagation (RUP) format [23, 22, 24]. In RUP, each inference step shows that a clause is logically entailed by the assumptions, by demonstrating that its negation leads to a contradiction through unit propagation. The proof concludes when falsity is derived, confirming unsatisfiability. In addition to clause-level validation, we have developed a TreeProof-level version of the checker, which constructs structured resolution traces that mirror the logical steps taken during unit propagation. These traces, called TreeProofs, represent the derivation of clauses from assumptions using a tree-like structure, where each node corresponds to either an assumption or a unit resolution step. This approach provides a structural view of the unit resolution process, enhancing confidence in the correctness of each inference. A TreeProof is valid if all assumptions referenced are within bounds and each resolution step is logically sound - particularly when resolving with unit clauses. To ensure soundness, we proved that if the checker returns true, then the assumptions logically entail the derived clause. By formalising and verifying both clause-level and TreeProof-level checking, we ensure that Z3's proofs can be trusted independently of the solver's internal mechanisms - an essential requirement for safety-critical applications.

Functions to check RUP inferences can be extracted from Rocq [37] into executable code using Rocq's extraction mechanism [32], typically to OCaml or Haskell. A parser for Z3 proof logs for ladder logic interlockings has also been developed. Extraction to other languages is possible, for example, C using the Codegen package [38]. Extraction to C supports basic types like numbers and lists, but complex types need extra handling or may not be supported. This is problematic for dependent types or higher-order functions lacking C equivalents.

Currently, proofs of correctness for RUP rely on generating all intermediate resolution proofs for each RUP inference. In fact, generating these proofs may be desirable when working with critical systems. Although having a proof that the checker is correct provides a high level of trust, there remains a remote possibility that an inconsistency in Rocq was used. Genuine bugs are occasionally detected in theorem provers. Therefore, having independently verifiable proof logs would allow for an even higher level of trust. The additional generated intermediate resolution proofs make it easier and therefore more trustworthy to verify the RUP proofs.

To support Z3 SAT proof logs, we extended our checker to validate Tseitin Transformations to convert non-CNF formulas for RUP whilst preserving satisfiability by introducing tautological clauses. Our prototype, implemented in Agda [39], matches these clauses against known patterns and formally proves their correctness [28]. This complements our verified RUP checker, forming a complete proof checker for propositional logic. This is now being replicated in OCaml for integration with the RUP Checker, with formal proofs in Rocq, ensuring each rule preserves satisfiability. By continuing to verify new rules in this way, we expand the checker's coverage while maintaining trustworthiness - essential for industrial applications.

References

[1] Michael Armand, Germain Faure, Benjamin Grégoire, Chantal Keller, Laurent Théry, and Benjamin Werner. A modular integration of sat/smt solvers to coq through proof witnesses. In Jean-Pierre Jouannaud and Zhong Shao, editors, *Certified Programs and Proofs*, pages 135–150, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[2] Madhusree Banerjee, Victor Cai, Sunitha Lakshmanappa, Andrew Lawrence, Markus Roggenbach, Monika Seisenberger, and Thomas Werner. A tool-chain for the verification of geographic scheme data. In Birgit Milius, Simon Collart-Dutilleul, and Thierry Lecomte, editors, *Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification*, pages 211–224, Cham, 2023. Springer Nature Switzerland. doi:10.1007/978-3-031-43366-5_13.

[3] Haniel Barbosa, Clark Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Ying Sheng, Cesare Tinelli, and Yoni Zohar. cvc5: A versatile and industrial-strength smt solver. In Dana Fisman and Grigore Rosu, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 415–442, Cham, 2022. Springer International Publishing.

[4] Haniel Barbosa, Jasmin Christian Blanchette, Mathias Fleury, Pascal Fontaine, and Hans-Jörg Schurr. Better SMT Proofs for Easier Reconstruction. In *AITP 2019 - 4th Conference on Artificial Intelligence and Theorem Proving*, Obergurgl, Austria, April 2019. URL: https://hal.science/hal-02381819.

[5] Haniel Barbosa, Andrew Reynolds, Gereon Kremer, Hanna Lachnitt, Aina Niemetz, Andres Nötzli, Alex Ozdemir, Mathias Preiner, Arjun Viswanathan, Scott Viteri, Yoni Zohar, Cesare Tinelli, and Clark Barrett. Flexible proof production in an industrial-strength smt solver. In Jasmin Blanchette, Laura Kovács, and Dirk Pattinson, editors, *Automated Reasoning*, pages 15–35, Cham, 2022. Springer International Publishing.

[6] Ulrich Berger, Andrew Lawrence, Fredrik Nordvall Forsberg, and Monika Seisenberger. Extracting verified decision procedures: DPLL and Resolution. *Logical Methods in Computer Science*, Volume 11, Issue 1, March 2015. URL: http://dx.doi.org/10.2168/LMCS-11(1:6)2015, doi:10.2168/lmcs-11(1:6)2015.

[7] Nikolaj Bjørner. Proofs for SMT, 11 October 2022. Slides, Dagstuhl, October 11 2022. URL: https://z3prover.github.io/slides/proofs.html#/.

[8] Sascha Böhme. Proof reconstruction for Z3 in Isabelle/HOL, 2009. Slides of talk given at 7th International Workshop on Satisfiability Modulo Theories (SMT '09). URL: https://wwwbroy.in.tum.de/~boehmes/proofrec-talk.pdf.

[9] Sascha Böhme. Proof reconstruction for Z3 in Isabelle/HOL. In *Workshop on Proof Exchange for Theorem Proving (PxTP)*, 2009. URL: https://www21.in.tum.de/~boehmes/proofrec.pdf.

[10] Sascha Böhme and Tjark Weber. Fast LCF-Style Proof Reconstruction for Z3. In Matt Kaufmann and Lawrence C. Paulson, editors, *Interactive Theorem Proving*, pages 179–194, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. doi:https://doi.org/10.1007/978-3-642-14052-5_14.

[11] Simon Chadwick, Phillip James, Markus Roggenbach, and Thomas Werner. Formal Methods for Industrial Interlocking Verification. In *ICIRT*, 2018. doi:10.1109/ICIRT.2018.8641579.

[12] Ran Chen, Cyril Cohen, Jean-Jacques Lévy, Stephan Merz, and Laurent Théry. Formal Proofs of Tarjan's Strongly Connected Components Algorithm in Why3, Coq and Isabelle. In John Harrison, John O'Leary, and Andrew Tolmach, editors, *10th International Conference on Interactive Theorem Proving (ITP 2019)*, volume 141 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:19, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ITP.2019.13, doi:10.4230/LIPIcs.ITP.2019.13.

[13] Alessio Coltellacci, Gilles Dowek, and Stephan Merz. Reconstruction of SMT proofs with Lambdapi. In Giles Reger and Yoni Zohar, editors, *CEUR Workshop Proceedings*, volume 3725, pages

13–23, Montréal, Canada, July 2024. URL: https://inria.hal.science/hal-04861898.

[14] Rocq community. About the Rocq Prover, Accessed 10 March 2025. URL: https://rocq-prover.org/about.

[15] 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. doi:10.1007/978-3-319-63046-5_14.

[16] Luís Cruz-Filipe, Joao Marques-Silva, and Peter Schneider-Kamp. Efficient Certified Resolution Proof Checking. In Axel Legay and Tiziana Margaria, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 118–135, Berlin, Heidelberg, 2017. Springer Berlin Heidelberg.

[17] Leonardo De Moura and Nikolaj Bjørner. Z3: an efficient SMT solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Proceedings of the Theory and Practice of Software, 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS'08/ETAPS'08, page 337–340, Berlin, Heidelberg, 2008. Springer-Verlag. URL: https://dl.acm.org/doi/abs/10.5555/1792734.1792766.

[18] SMTCoq Developers. Smtcoq: Communication between coq and sat/smt solvers. https://smtcoq.github.io/, 2025. Accessed: 2025-06-26.

[19] Nick Feng, Alan J. Hu, Sam Bayless, Syed M. Iqbal, Patrick Trentin, Mike Whalen, Lee Pike, and John Backes. DRAT proofs of unsatisfiability for SAT Modulo Monotonic theories, 2024. URL: https://arxiv.org/abs/2401.10703, arXiv:2401.10703.

[20] Mathias Fleury and Peter Lammich. A more pragmatic cdcl for isasat and targetting llvm (short paper). In Brigitte Pientka and Cesare Tinelli, editors, *Automated Deduction – CADE 29*, pages 207–219, Cham, 2023. Springer Nature Switzerland.

[21] Mathias Fleury and Hans-Jörg Schurr. Reconstructing veriT proofs in Isabelle/HOL. *Electronic Proceedings in Theoretical Computer Science*, 301:36–50, August 2019. doi:10.4204/eptcs.301.6.

[22] Allen van Gelder. Verifying RUP proofs of Propositional Unsatisfiability, 2008. Slides of a talk given at ISAIM'08. URL: https://users.soe.ucsc.edu/~avg/ProofChecker/Documents/proofs-isaim08-trans.pdf.

[23] Allen van Gelder. Verifying RUP Proofs of Propositional Unsatisfiability: Have Your Cake and Eat It Too, 2008. In Proceedings of 10th International Symposium on Artificial Intelligence and Mathematics (ISAIM'08). URL: https://users.soe.ucsc.edu/~avg/ProofChecker/Documents/proofs-isaim08-long.pdf.

[24] Eugene Goldberg and Yakov Novikov. Verification of Proofs of Unsatisfiability for CNF Formulas. In *2003 Design, Automation and Test in Europe Conference and Exhibition*, pages 886–891, March 2003. doi:10.1109/DATE.2003.1253718.

[25] CMU Transparency Group. verified_rup: A verified drup and drat proof checker. https://github.com/cmu-transparency/verified_rup, 2023. Accessed: 2025-06-26.

[26] Alan G. Hamilton. *Logic for Mathematicians*. Cambridge University Press, Cambridge, rev. ed. edition, 1988.

[27] Matti Järvisalo, Marijn J. H. Heule, and Armin Biere. Inprocessing Rules. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *Automated Reasoning*, pages 355–370, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. doi:https://doi.org/10.1007/978-3-642-31365-3_28.

[28] Markus Krötzsch. *Description Logic Rules*. IOS Press, 2010.

[29] Elias Kuiter, Sebastian Krieter, Chico Sundermann, Thomas Thüm, and Gunter Saake. Tseitin or not Tseitin? The Impact of CNF Transformations on Feature-Model Analyses. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, ASE '22, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3551349.3556938.

[30] Peter Lammich. Fast and verified unsat certificate checking. In Christoph Benzmüller, Marijn J.H. Heule, and Renate A. Schmidt, editors, *Automated Reasoning*, pages 439–457, Cham,

2024. Springer Nature Switzerland.

[31] S. Lescuyer and S. Conchon. A Reflexive Formalization of a SAT Solver in Coq, 2008. In Otmane Ait Mohamed, César Muñoz, and Sofiène Tahar: 21st International Conference on Theorem Proving in Higher Order Logics. Proceedings of TPHOL08. Technical Report (2008-1-Ait Mohamed), pages 65 - 76. Available from https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=971acf742e8a2d29a56cf46343320966b18fff86.

[32] Pierre Letouzey. Extraction in Coq, an Overview. In Arnold Beckmann, Costas Dimitracopoulos, and Benedikt Löwe, editors, *Logic and Theory of Algorithms: 4th Conference on Computability in Europe, CiE 2008, Athens, Greece, June 15-20, 2008 Proceedings 4*, pages 359–369, June 2008. doi:10.1007/978-3-540-69407-6_39.

[33] F. Marić and P. Janičić. Formal Correctness Proof for DPLL Procedure. *Informatica*, 21(1):57–78, 2010.

[34] Marius Minea. Conjunctive Normal Form: Tseitin Transform, 2024. H250: Honors Colloquium – Introduction to Computation. URL: https://people.cs.umass.edu/~marius/class/h250/lec2.pdf.

[35] Duckki Oe, Aaron Stump, Corey Oliver, and Kevin Clancy. versat: A Verified Modern SAT Solver. In Viktor Kuncak and Andrey Rybalchenko, editors, *Verification, Model Checking, and Abstract Interpretation*, pages 363–378, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[36] Adrián Rebola-Pardo. Even Shorter Proofs Without New Variables. In Meena Mahajan and Friedrich Slivovsky, editors, *26th International Conference on Theory and Applications of Satisfiability Testing (SAT 2023)*, volume 271 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:20, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.SAT.2023.22, doi:10.4230/LIPIcs.SAT.2023.22.

[37] Rocq Development Team. Rocq. https://rocq-prover.org/, 2025. Accessed: 2025-05-09.

[38] Akira Tanaka. Coq to C translation with partial evaluation. In *Proceedings of the 2021 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation*, PEPM 2021, pages 14–31, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3441296.3441394.

[39] The Agda Team. What is agda? https://agda.readthedocs.io/en/latest/getting-started/what-is-agda.html, 2025. Accessed: 2025-06-24.

[40] Grigori S. Tseitin. On the complexity of derivation in propositional calculus. In Jörg H. Siekmann and Graham Wrightson, editors, *Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970*, pages 466–483, Berlin, Heidelberg, 1983. Springer Berlin Heidelberg. doi:10.1007/978-3-642-81955-1_28.

[41] Nathan Wetzler, Marijn Heule, and Warren Hunt. DRAT-trim: Efficient Checking and Trimming Using Expressive Clausal Proofs. In *International Conference on Theory and Applications of Satisfiability Testing*, 07 2014. doi:10.1007/978-3-319-09284-3_31.

[42] Nathan Wetzler, Marijn J. H. Heule, and Warren A. Hunt. Mechanical verification of SAT refutations with extended resolution. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *Interactive Theorem Proving*, pages 229–244, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. URL: https://doi.org/10.1007/978-3-642-39634-2_18, doi:https://doi.org/10.1007/978-3-642-39634-2_18.

[43] Nathan David Wetzler. *Efficient, mechanically-verified validation of satisfiability solvers*. PhD thesis, Univeristy of Texas at Austin, Austin, 2015. URL: https://repositories.lib.utexas.edu/bitstream/handle/2152/30538/WETZLER-DISSERTATION-2015.pdf?sequence=1.