A Verified SAT Solver Framework with Learn, Forget, Restart, and Incrementality (Extended Abstract)

Jasmin Christian Blanchette, Mathias Fleury, Christoph Weidenbach

A Verified SAT Solver Framework with Learn, Forget, Restart, and Incrementality (Extended Abstract)

Jasmin Christian Blanchette[1,2], Mathias Fleury[2], and Christoph Weidenbach[2]

[1] Inria Nancy – Grand Est & LORIA, Villers-lès-Nancy, France
[2] Max-Planck-Institut für Informatik, Saarbrücken, Germany

Abstract. We developed a formal framework for CDCL (conflict-driven clause learning) in Isabelle/HOL. Through a chain of refinements, an abstract CDCL calculus is connected to a SAT solver expressed in a functional programming language, with total correctness guarantees. The framework offers a convenient way to prove metatheorems and experiment with variants. Compared with earlier SAT solver verifications, the main novelties are the inclusion of rules for forget, restart, and incremental solving and the application of refinement.

Researchers in automated reasoning spend a significant portion of their work time specifying logical calculi and proving metatheorems about them. These proofs are typically carried out with pen and paper, which is error-prone and can be tedious. As proof assistants are becoming easier to use, it makes sense to employ them.

In this spirit, we started an effort, called IsaFoL (Isabelle Formalization of Logic), that aims at developing libraries and methodology for formalizing modern research in the field, using the Isabelle/HOL proof assistant [5]. Our initial emphasis is on established results about propositional and first-order logic. In particular, we are formalizing large parts of Weidenbach's forthcoming textbook, tentatively called *Automated Reasoning—The Art of Generic Problem Solving*. Our inspiration for formalizing logic is the IsaFoR project, which focuses on term rewriting [14].

The objective of formalization work is not to eliminate paper proofs, but to complement them with rich formal companions. Formalizations help catch mistakes, whether superficial or deep, in specifications and theorems; they make it easy to experiment with changes or variants of concepts; and they help clarify concepts left vague on paper.

We present our formalization of CDCL from *Automated Reasoning* on propositional satisfiability (SAT), developed via a refinement of Nieuwenhuis, Oliveras, and Tinelli's account of CDCL [13]. CDCL is the algorithm implemented in modern SAT solvers. We start with a family of abstract DPLL [6] and CDCL [2, 3, 10, 12] transition systems. Some of the calculi include rules for learning and forgetting clauses and for restarting the search. All calculi are proved sound and complete, as well as terminating under a reasonable strategy. The abstract CDCL calculus is refined into the more concrete calculus presented in *Automated Reasoning* and recently published [16]. The latter specifies a criterion for learning clauses representing first unit implication points (1UIPs) [3], with the guarantee that learned clauses are not redundant and hence derived at most

once. The calculus also supports incremental solving. This concrete calculus is refined further, as a certified functional program extracted using Isabelle's code generator.

Any formalization effort is a case study in the use of a proof assistant. Beyond the code generator, we depended heavily on the following features of Isabelle:

- *Isar* [17] is a textual proof format inspired by the pioneering Mizar system [11]. It makes it possible to write structured, readable proofs—a requisite for any formalization that aims at clarifying an informal proof.
- *Locales* [1, 9] parameterize theories over operations and assumptions, encouraging a modular style of development. They are useful to express hierarchies of related concepts and to reduce the number of parameters and assumptions that must be threaded through a formal development.
- *Sledgehammer* integrates superposition provers and SMT (satisfiability modulo theories) solvers in Isabelle to discharge proof obligations. The SMT solvers, and one of the superposition provers [15], are built around a SAT solver, resulting in a situation where SAT solvers are employed to prove their own metatheory.

Our work is related to other verifications of SAT solvers, typically with the aim of increasing their trustworthiness. This goal has lost some of its significance with the emergence of formats for certificates that are easy to generate, even in highly optimized solvers, and that can be processed efficiently by verified checkers [8]. In contrast, our focus is on formalizing the metatheory of CDCL, to study and connect the various members of the family. The main novelties of our framework are the inclusion of rules for forget, restart, and incremental solving and the application of refinement to transfer results. The framework is available online as part of the IsaFoL repository [7].

This extended abstract is based on our paper [4] presented at IJCAR 2016. It is available online.[1]

References

[1] Ballarin, C.: Locales: A module system for mathematical theories. J. Autom. Reasoning 52(2), 123–153 (2014)

[2] Bayardo Jr., R.J., Schrag, R.: Using CSP look-back techniques to solve exceptionally hard SAT instances. In: Freuder, E.C. (ed.) CP96. LNCS, vol. 1118, pp. 46–60. Springer (1996)

[3] Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press (2009)

[4] Blanchette, J.C., Fleury, M., Weidenbach, C.: A verified SAT solver framework with learn, forget, restart, and incrementality. In: Olivetti, N., Tiwari, A. (eds.) IJCAR 2016. LNCS, vol. 9706. Springer (2016)

[5] Blanchette, J.C., Fleury, M., Schlichtkrull, A., Traytel, D.: IsaFoL: Isabelle Formalization of Logic, `https://bitbucket.org/jasmin_blanchette/isafol`

[6] Davis, M., Logemann, G., Loveland, D.W.: A machine program for theorem-proving. Commun. ACM 5(7), 394–397 (1962)

[7] Fleury, M., Blanchette, J.C.: Formalization of Weidenbach's *Automated Reasoning—The Art of Generic Problem Solving*, `https://bitbucket.org/jasmin_blanchette/isafol/src/master/Weidenbach_Book/README.md`

[1] `http://www.loria.fr/~jablanch/sat.pdf`

[8] Heule, M., Hunt Jr., W.A., Wetzler, N.: Bridging the gap between easy generation and efficient verification of unsatisfiability proofs. Softw. Test. Verif. Reliab. 24(8), 593–607 (2014)
[9] Kammüller, F., Wenzel, M., Paulson, L.C.: Locales—A sectioning concept for Isabelle. In: Bertot, Y., Dowek, G., Hirschowitz, A., Paulin, C., Théry, L. (eds.) TPHOLs '99. LNCS, vol. 1690, pp. 149–166. Springer (1999)
[10] Marques-Silva, J.P., Sakallah, K.A.: GRASP—A new search algorithm for satisfiability. In: ICCAD '96. pp. 220–227. IEEE Computer Society Press (1996)
[11] Matuszewski, R., Rudnicki, P.: Mizar: The first 30 years. Mechanized Mathematics and Its Applications 4(1), 3–24 (2005)
[12] Moskewicz, M.W., Madigan, C.F., Zhao, Y., Zhang, L., Malik, S.: Chaff: Engineering an efficient SAT solver. In: DAC 2001. pp. 530–535. ACM (2001)
[13] Nieuwenhuis, R., Oliveras, A., Tinelli, C.: Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T). J. ACM 53(6), 937–977 (2006)
[14] Sternagel, C., Thiemann, R.: An Isabelle/HOL formalization of rewriting for certified termination analysis, http://cl-informatik.uibk.ac.at/software/ceta/
[15] Voronkov, A.: AVATAR: The architecture for first-order theorem provers. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 696–710. Springer (2014)
[16] Weidenbach, C.: Automated reasoning building blocks. In: Meyer, R., Platzer, A., Wehrheim, H. (eds.) Correct System Design: Symposium in Honor of Ernst-Rüdiger Olderog on the Occasion of His 60th Birthday. LNCS, vol. 9360, pp. 172–188. Springer (2015)
[17] Wenzel, M.: Isabelle/Isar—A generic framework for human-readable proof documents. In: Matuszewski, R., Zalewska, A. (eds.) From Insight to Proof: Festschrift in Honour of Andrzej Trybulec, Studies in Logic, Grammar, and Rhetoric, vol. 10(23). University of Białystok (2007)