EPiC Series in Computing
Volume 38, 2016, Pages 1–7
Proceedings of the 1st and 2nd Vampire Workshops

My Life with an Automatic Theorem Prover

Jasmin Christian Blanchette
Technische Universität München, Germany

Abstract

Sledgehammer integrates third-party automatic theorem provers in the proof assistant Isabelle/HOL. In the seven years since its first release in 2007, it has grown to become an essential part of most Isabelle users' workflow. Although a lot of effort has gone into tuning the system, the main reason for Sledgehammer's success is the impressive power of the external provers, especially E, SPASS, Vampire, and Z3. In this paper, I review Vampire's strengths and weaknesses in this context and propose a few directions for future work.

Isabelle/HOL [27] is a proof assistant based on polymorphic higher-order logic extended with axiomatic type classes. Its tool Sledgehammer [5,28] integrates third-party automatic theorem provers to increase the level of automation. It consists of three main components:

- The *relevance filter* [23,26] heuristically selects a few hundred facts (lemmas, definitions, axioms) from the thousands available to Isabelle users.
- The *translation module* [6,25] encodes polymorphic higher-order propositions in the target prover's logic (e.g., untyped or monomorphic first-order logic).
- The *reconstruction module* [29, 33] produces a textual proof that can be inserted in an Isabelle development to replay the proof.

Sledgehammer was originally designed by Lawrence Paulson and his team from 2003 to 2009. I took over its development in 2010 at Tobias Nipkow's suggestion. In the last seven years or so, it has grown to become an essential part of most Isabelle users' workflow.

The following provers are supported as backends:

- *Unit-equality prover*: Waldmeister [17];
- *Superposition-based first-order provers*: E [32], SPASS [38], and Vampire [31];
- *Instantiation-based first-order provers*: iProver [19] and iProver-Eq [20];
- *SMT* (*satisfiability modulo theories*) *solvers*: Alt-Ergo [10], CVC3 [3], Yices [14], and Z3 [13];
- *Higher-order provers*: AgsyHOL [24], LEO-II [4], and Satallax [12].

It is straightforward to extend this list with provers that can communicate in the TPTP/TSTP [35] or SMT-LIB [30] formats. For historical reasons, the SMT solver translation is distinct from the TPTP/TSTP translation. Originally, most provers developed in the community around CADE and TPTP had no support for types or arithmetic, but this is now changing [37].

Systems such as Isabelle adhere to the tradition initiated in the 1970s by the LCF system [16]: All inferences are derived by a small trusted kernel; types and functions are defined rather than axiomatized to guard against inconsistencies. My experience has confirmed the value of the foundational–definitional approach: In the past five years, I or my tools have found soundness bugs in seven external provers (Alt-Ergo, LEO-II, TPS, Vampire, Yices, Z3, and Zipperposition), but not in Isabelle or any of the other LCF-style provers. For this reason, reconstruction is an essential component of Sledgehammer.

Reconstruction is performed by generating a piece of Isabelle proof text that can replay the proof without depending on the external prover. In most cases, a single call to the built-in resolution prover Metis [18, 29] suffices to re-find the proof in Isabelle. In ongoing work, my colleagues and I are extending Sledgehammer to generate detailed, structured Isabelle proofs based on the external proofs [33]. Inferences are justified by suitable proof methods.

As an added benefit, the proofs can be inspected and understood by the user. In principle, the raw proofs generated by the external provers can also be inspected. In practice, the reconstructed Isabelle proofs are much more readable, for several reasons: They are in terms of the original logic and not of the encoding in the target logic; they are direct instead of by contradiction; and they collapse straightforward chains of reasoning into single inferences.

By default, Sledgehammer runs E, SPASS, Vampire, and Z3 in parallel. Vampire is run remotely on the SystemOnTPTP service [34], unless the user has installed it locally. The other provers are part of the Isabelle distribution and run locally in separate threads. The default time limit is 30 seconds per thread. Even though E and Vampire implement strategy scheduling, Sledgehammer performs its own scheduling, generating a sequence of problems with different numbers of facts, different encodings of types, and different prover options.

Vampire is famous in the automated reasoning community for its repeated victories at the CADE ATP System Competition (CASC) [36]. On Sledgehammer-generated benchmarks, the race between E, SPASS, Vampire, and Z3 is very close [5, 9, 11, 23, 28]—but my latest, unpublished benchmarks [7] put Vampire 3.0 clearly ahead of its competitors.

Among Vampire's many features, the support for monomorphic types (as defined by TPTP TFF0 [37]) and the detailed TSTP proofs are especially beneficial to Sledgehammer:

- *The monomorphic types, in conjunction with a heuristic monomorphizer, make it easy to encode Isabelle's type information in a natural, lightweight fashion.* Other encodings in terms of untyped first-order logic are possible, but none are as efficient as the native support in Vampire [7].
- *Vampire can output detailed, inference-level proofs in TSTP format.* Some options must be disabled to make the proofs intelligible, but once this is done the proofs can normally be reconstructed inference by inference in Isabelle using Metis. Unlike some of its rivals, Vampire records the entire clausification and skolemization process in its proof output, making it possible to use the prover as a backend in a larger tool in accordance with what

I call the *Russian doll principle*.[1]

In recent years, some work has been done on adding arithmetic to Vampire [21, 22]. We have yet to try out this feature. Our experience with arithmetic in SMT solvers is that it helps increase the success rate by a few percentage points, without being an absolutely essential feature [5]. Isabelle's libraries are full of arithmetic lemmas that can be used in the absence of dedicated theory reasoning in the prover.

Another promising feature of Vampire is its (currently unpublished) detection and optimization of extensionality reasoning. In Isabelle, function equality is extensional, and most Sledgehammer problems contain axioms such as $\forall F, G.\ (\forall X.\ \mathsf{app}(F, X) \approx \mathsf{app}(G, X)) \longrightarrow F \approx G$, where $\mathsf{app}(F, X)$ denotes the application of F to X. Virtually all provers have difficulties applying such axioms when they are needed, because of the highly general literal $F \approx G$.

There are a number of areas in which I would welcome further development of Vampire. They are reported here in a friendly, constructive spirit. Quite a few issues revolve around basic system engineering:

- *Vampire's development takes place behind closed doors.* The cycle of bug reports and fixes can take months if not years. When types were introduced in Vampire, I immediately found a soundness bug and reported it. The Vampire developers were quick at fixing the bug, but I had to wait until the next release to try out the fix, only to discover another type-related bug. A more agile development model would have helped make this new feature converge more rapidly.
- *The prover's command-line options and inference rules are not well documented.* Some information can be found in outdated slides online. At one point, I was given access to a Google Docs page that documented the options, but the link is now broken.
- *The restrictive terms of the prover's license means that we cannot distribute Vampire as part of the Isabelle distribution.* Users can download it themselves and install it locally, but the vast majority never gets this far. Fortunately, the presence of Vampire on SystemOnTPTP means that these users benefit from the prover, at least when they are not sitting in a metro or an airplane.
- *It is difficult to obtain the latest version of the tool.* The official website is desperately out of date. As a result, I usually find myself using the source version submitted to CASC, despite the forbidding wording in the source file headers:

> *Posession [sic] of, or access to, this program does not give you or anybody else a right or licence to distribute, modify, copy, compile, create derivatives, or use in any form or for any purpose, this program, or any part thereof.*
> *Any licence to use Vampire shall only be obtained as described on Vampire's home page* `http://vprover.org`*.*

The above issues should be relatively easy to address, and I have no doubts that they would contribute to increase Vampire's deployment in academia and industry. In this respect, the

[1] In contrast, SPASS proofs are expressed in terms of the clausified problem. Proof reconstruction is possible only with reverse-engineering of the various transformations performed by SPASS's clausifier [2].

commercially developed Z3 offers an interesting model: It combines commercial software development imperatives (including the expensive commercial licenses) with some of the best practices from the open source community: The source repository is publicly available, executables are built every night for FreeBSD, Linux, Mac OS X, and Windows, and there is a vibrant community that discusses bugs and feature requests online.

On the automated reasoning front, the following enhancements could help Sledgehammer:

- *Vampire could support rank-1 polymorphism.* The TPTP syntax has recently been extended with a suitable format for proofs and solutions, called TFF1 [8]. The heuristic monomorphization performed by Sledgehammer to bridge the gap between Isabelle's polymorphic logic and Vampire's monomorphic logic is incomplete and can dramatically increase the size of the problem. Although I have no conclusive empirical evidence yet, this grounding at the type level is almost certainly suboptimal compared with native support for polymorphism in the prover.
- *It would be useful to have a simple technology to tune the prover based on a custom set of benchmarks.* Vampire has a large number of parameters and strategies, but these are tuned against problems from the TPTP library. For example, on an eight-core machine, Sledgehammer might want to run several instances of Vampire in parallel, with complementary strategies—but having to come up with the right setup for these Vampire instances without basic tool support is not an attractive proposal.

In a collaboration with the SPASS developers, we are looking at ways to further reduce the gap between automatic theorem provers and proof assistants [9]. This research is ongoing. So far, SPASS's superposition calculus has been extended with linear and nonlinear arithmetic [1, 15], a polymorphic type system with type classes, datatype reasoning, and (co)induction. The ideas have been implemented in various SPASS clones and prototypes and are expected to be part of the forthcoming version 4.0. Similar ideas could be implemented in Vampire.

A promising avenue of work that could help derive more difficult proofs would be to have the automatic theorem prover (partially) pre-saturate the background theories and store the resulting inferences across invocations, instead of re-deriving the same consequences over and over again. This requires a uniform, compositional problem encoding—a realistic prospect once the external prover supports polymorphism.

Acknowledgment. A lot of people have contributed to Sledgehammer over the years, including Sascha Böhme, Jia Meng, Tobias Nipkow, Lawrence Paulson, and Makarius Wenzel. The Vampire developers, especially Kryštof Hoder and Andrei Voronkov, have helped us get the most out of their most remarkable tool. Ondřej Kunčar and Tobias Nipkow commented on a draft of this paper. Mark Summerfield suggested several textual improvements. My research is financed by the Deutsche Forschungsgemeinschaft (DFG) project Hardening the Hammer (grant Ni 491/14-1).

References

[1] E. Althaus, E. Kruglov, and C. Weidenbach. Superposition modulo linear arithmetic SUP(LA). In S. Ghilardi and R. Sebastiani, eds., *FroCoS 2009*, vol. 5749 of *LNCS*, pp. 84–99. Springer, 2009.

[2] N. Azmy and C. Weidenbach. Computing tiny clause normal forms. In M. P. Bonacina, ed., *CADE-24*, vol. 7898 of *LNCS*, pp. 109–125. Springer, 2013.

[3] C. Barrett and C. Tinelli. CVC3. In W. Damm and H. Hermanns, eds., *CAV 2007*, vol. 4590 of *LNCS*, pp. 298–302. Springer, 2007.

[4] C. Benzmüller, L. C. Paulson, F. Theiss, and A. Fietzke. LEO-II—A cooperative automatic theorem prover for higher-order logic. In A. Armando, P. Baumgartner, and G. Dowek, eds., *IJCAR 2008*, vol. 5195 of *LNAI*, pp. 162–170. Springer, 2008.

[5] J. C. Blanchette, S. Böhme, and L. C. Paulson. Extending Sledgehammer with SMT solvers. In N. Bjørner and V. Sofronie-Stokkermans, eds., *CADE-23*, vol. 6803 of *LNAI*, pp. 207–221. Springer, 2011.

[6] J. C. Blanchette, S. Böhme, A. Popescu, and N. Smallbone. Encoding monomorphic and polymorphic types. In N. Piterman and S. Smolka, eds., *TACAS 2013*, vol. 7795 of *LNCS*, pp. 493–507. Springer, 2013.

[7] J. C. Blanchette, S. Böhme, A. Popescu, and N. Smallbone. Encoding monomorphic and polymorphic types. http://www21.in.tum.de/~blanchet/enc_types_article.pdf. Extended version of TACAS 2013 paper [6]. Submitted.

[8] J. C. Blanchette and A. Paskevich. TFF1: The TPTP typed first-order form with rank-1 polymorphism. In M. P. Bonacina, ed., *CADE-24*, vol. 7898 of *LNAI*, pp. 414–420. Springer, 2013.

[9] J. C. Blanchette, A. Popescu, D. Wand, and C. Weidenbach. More SPASS with Isabelle: Superposition with hard sorts and configurable simplification. In L. Beringer and A. Felty, eds., *ITP 2012*, vol. 7406 of *LNCS*, pp. 345–360. Springer, 2012.

[10] F. Bobot, S. Conchon, E. Contejean, and S. Lescuyer. Implementing polymorphism in SMT solvers. In C. Barrett and L. de Moura, eds., *SMT '08*, ICPS, pp. 1–5. ACM, 2008.

[11] S. Böhme and T. Nipkow. Sledgehammer: Judgement Day. In J. Giesl and R. Hähnle, eds., *IJCAR 2010*, vol. 6173 of *LNAI*, pp. 107–121. Springer, 2010.

[12] C. E. Brown. Satallax: An automatic higher-order prover. In B. Gramlich, D. Miller, and U. Sattler, eds., *IJCAR 2012*, vol. 7364 of *LNCS*, pp. 111–117. Springer, 2012.

[13] L. de Moura and N. Bjørner. Z3: An efficient SMT solver. In C. R. Ramakrishnan and J. Rehof, eds., *TACAS 2008*, vol. 4963 of *LNCS*, pp. 337–340. Springer, 2008.

[14] B. Dutertre and L. de Moura. The Yices SMT solver. http://yices.csl.sri.com/tool-paper.pdf, 2006.

[15] A. Eggers, E. Kruglov, S. Kupferschmid, K. Scheibler, T. Teige, and C. Weidenbach. Superposition modulo non-linear arithmetic. In C. Tinelli and V. Sofronie-Stokkermans, eds., *FroCoS 2011*, vol. 6989 of *LNCS*, pp. 119–134. Springer, 2011.

[16] M. J. C. Gordon, R. Milner, and C. P. Wadsworth. *Edinburgh LCF: A Mechanised Logic of Computation*, vol. 78 of *LNCS*. Springer, 1979.

[17] T. Hillenbrand, A. Buch, R. Vogt, and B. Löchner. WALDMEISTER—High-performance equational deduction. *J. Autom. Reasoning*, 18(2):265–270, 1997.

[18] J. Hurd. First-order proof tactics in higher-order logic theorem provers. In M. Archer, B. Di Vito, and C. Muñoz, eds., *Design and Application of Strategies/Tactics in Higher Order Logics*, NASA

Tech. Reports, pp. 56–68, 2003.

[19] K. Korovin. Instantiation-based automated reasoning: From theory to practice. In R. A. Schmidt, ed., *CADE-22*, vol. 5663 of *LNAI*, pp. 163–166. Springer, 2009.

[20] K. Korovin and C. Sticksel. iProver-Eq: An instantiation-based theorem prover with equality. In J. Giesl and R. Hähnle, eds., *IJCAR 2010*, vol. 6173 of *LNCS*, pp. 196–202. Springer, 2010.

[21] K. Korovin and A. Voronkov. Integrating linear arithmetic into superposition calculus. In J. Duparc and T. A. Henzinger, eds., *CSL 2007*, vol. 4646 of *LNCS*, pp. 223–237. Springer, 2007.

[22] K. Korovin and A. Voronkov. Solving systems of linear inequalities by bound propagation. In N. Bjørner and V. Sofronie-Stokkermans, eds., *CADE-23*, vol. 6803 of *LNCS*, pp. 369–383. Springer, 2011.

[23] D. Kühlwein, J. C. Blanchette, C. Kaliszyk, and J. Urban. MaSh: Machine learning for Sledgehammer. In S. Blazy, C. Paulin-Mohring, and D. Pichardie, eds., *ITP 2013*, vol. 7998 of *LNCS*, pp. 35–50. Springer, 2013.

[24] F. Lindblad. A focused sequent calculus for higher-order logic. In S. Demri, D. Kapur, and C. Weidenbach, eds., *IJCAR 2014*, LNAI. Springer, 2014.

[25] J. Meng and L. C. Paulson. Translating higher-order clauses to first-order clauses. *J. Autom. Reasoning*, 40(1):35–60, 2008.

[26] J. Meng and L. C. Paulson. Lightweight relevance filtering for machine-generated resolution problems. *J. Applied Logic*, 7(1):41–57, 2009.

[27] T. Nipkow, L. C. Paulson, and M. Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, vol. 2283 of *LNCS*. Springer, 2002.

[28] L. C. Paulson and J. C. Blanchette. Three years of experience with Sledgehammer, a practical link between automatic and interactive theorem provers. In G. Sutcliffe, S. Schulz, and E. Ternovska, eds., *IWIL-2010*, vol. 2 of *EPiC*, pp. 1–11. EasyChair, 2012.

[29] L. C. Paulson and K. W. Susanto. Source-level proof reconstruction for interactive theorem proving. In K. Schneider and J. Brandt, eds., *TPHOLs 2007*, vol. 4732 of *LNCS*, pp. 232–245. Springer, 2007.

[30] S. Ranise and C. Tinelli. The SMT-LIB standard: Version 1.2. `http://goedel.cs.uiowa.edu/smtlib/papers/format-v1.2-r06.08.30.pdf`, 2006.

[31] A. Riazanov and A. Voronkov. The design and implementation of Vampire. *AI Comm.*, 15(2-3):91–110, 2002.

[32] S. Schulz. System description: E 1.8. In K. McMillan, A. Middeldorp, and A. Voronkov, eds., *LPAR-19*, vol. 8312 of *LNAI*, pp. 735–743. Springer, 2013.

[33] S. J. Smolka and J. C. Blanchette. Robust, semi-intelligible Isabelle proofs from ATP proofs. In J. C. Blanchette and J. Urban, eds., *PxTP 2013*, vol. 14 of *EPiC*, pp. 117–132. EasyChair, 2013.

[34] G. Sutcliffe. System description: SystemOnTPTP. In D. McAllester, ed., *CADE-17*, vol. 1831 of *LNAI*, pp. 406–410. Springer, 2000.

[35] G. Sutcliffe. The TPTP problem library and associated infrastructure: The FOF and CNF parts, v3.5.0. *J. Autom. Reasoning*, 43(4):337–362, 2009.

[36] G. Sutcliffe. The 6th IJCAR automated theorem proving system competition—CASC-J6. *AI Comm.*, 26(2):211–223, 2013.

[37] G. Sutcliffe, S. Schulz, K. Claessen, and P. Baumgartner. The TPTP typed first-order form with arithmetic. In N. Bjørner and A. Voronkov, eds., *LPAR-18*, vol. 7180 of *LNCS*, pp. 406–419. Springer, 2012.

[38] C. Weidenbach, D. Dimova, A. Fietzke, R. Kumar, M. Suda, and P. Wischnewski. SPASS version 3.5. In R. A. Schmidt, ed., *CADE-22*, vol. 5663 of *LNAI*, pp. 140–145. Springer, 2009.