A Lambda-Superposition Tactic for Isabelle/HOL (Extended Abstract)

Massin Guerdi

Ludwig-Maximilians-Universität München, Germany
massin.guerdi@lmu.de

Abstract. I introduce *slam*, an Isabelle/HOL tactic and automated theorem prover based on the λ-superposition calculus. An alternative to Isabelle's *metis* tactic, *slam* targets higher-order logic directly, avoiding the overhead introduced by *metis*' translations to first-order logic. Like *metis*, *slam* can be used as a reconstruction backend for Sledgehammer to improve the reconstruction rate of proofs produced by external higher-order automated theorem provers such as E, Vampire, and Zipperposition.

Sledgehammer [10] is a proof tool for Isabelle/HOL [8] that integrates external first- and higher-order automated theorem provers (ATPs). For an external proof to be accepted by Isabelle, it needs to be reconstructed using Isabelle's inference kernel. One widely used proof reconstruction method is the *metis* [11] tactic, which is based on the Metis [6] ATP. A weakness of *metis* is that it relies on potentially inefficient, incomplete translations from polymorphic higher-order logic to Metis' untyped first-order logic. As a result, reconstruction can fail for proofs found by higher-order provers such as E [12, 16], Vampire [3, 7], and Zipperposition [1, 5]. To solve this issue, I introduce *slam*,[1] a tactic that natively reasons about higher-order logic.

Architecture The *slam* tactic consists of an ATP which performs proof search and a proof reconstruction module. This split into two phases is also used by the *metis* and *blast* [9] tactics. It allows the proof search to be more flexible, as it does not need to justify the steps it performs and can make use of efficient data structures. Once a proof has been found, reconstruction takes over and replays only that part of the search that is needed for the proof. Still, to ensure as good a reconstruction rate as possible, *slam* reuses many parts of Isabelle's implementation during the proof search, like the term data structure and unification algorithm.

Calculus Like E, Vampire, and Zipperposition, *slam* implements a variant of λ-superposition—a generalization of the first-order superposition calculus to higher-order logic. Superposition calculi are refutational: they work by assuming the negation of the goal and deriving a contradiction. The λ-superposition calculi [1, 2] are refutationally complete with respect to Henkin semantics. In practice, however, provers have more success using incomplete, *pragmatic* variants, which try to tame the combinatorial explosions inherent to higher-order logic [2, Fig. 5].

[1] https://github.com/mguerdi/slam

Booleans The calculus implemented by *slam* is based on λ-superposition with booleans (oλSup) [1]. A clause in oλSup is a disjunction of equality literals. Since the calculus has first-class support for booleans, it is not necessary to fully clausify the input problem. Instead, *slam* relies on certain inference and simplification rules to perform stepwise clausification during the proof search.

Proof Search Following Schulz [12], *slam* uses the DISCOUNT variant of the given-clause procedure. Unlike Zipperposition's complete modes, where some inferences generate an infinite number of conclusions, *slam* only ever considers a finite number, discarding the rest. It also uses Isabelle's incomplete higher-order unification procedure. To simplify and prune the search state, *slam* implements most of the simplifications from Schulz [12]. It uses fingerprint indexing [13, 15] and feature-vector indexing [14, 16] to speed up inferences and simplifications.

Proof Reconstruction The initial setup of the tactic provides lemmas passed in by the user, assumptions in the proof state, and the negated conjecture as elements of Isabelle's theorem datatype. The reconstruction module must then produce a theorem carrying a proof of `False` to close the goal. Each clause derived by *slam* during proof search carries an `origin` that records the premises, how they were instantiated, and which inference rule was used. This information is used to reconstruct all ancestors of the contradiction, in chronological order, storing each derived theorem in a cache.

Skolemization A noteworthy aspect of proof reconstruction is the treatment of inference rules that introduce Skolem constants. Using a similar approach to Böhme [4], *slam*'s proof reconstruction realizes Skolem constants free variables $\mathtt{sk}_i$. To logically justify the introduction of a Skolem constant $\mathtt{sk}_i$ in a reconstructed clause, an extra premise is added to the clause, which defines $\mathtt{sk}_i$ as a function $\lambda y.\, \lambda x_1. \ldots \lambda x_n.\, \varepsilon z.\, P\, z$ that returns the corresponding witness, using Hilbert's ε operator. Each x_j abstracts over one of the schematic variables occurring in the expression P being skolemized. The variable y is unused in the body, but has a dummy type that accounts for hidden polymorphism. The last step of proof reconstruction is to eliminate these premises by instantiating the $\mathtt{sk}_i$'s with the right-hand side of their defining equation, at which point the extra premises become true by reflexivity.

Evaluation To evaluate *slam*, it was integrated as a proof reconstruction backend into Sledgehammer and compared to *metis* on 1127 evenly distributed goals from 49 randomly chosen entries from the Archive of Formal Proofs[2]. The goals were chosen by running Sledgehammer with Zipperposition, which, for each successful goal, yielded a set of lemmas that were then passed to *metis* and *slam*.

Of these, 1011 goals were successfully solved by some variant of *metis*, while *slam* solved 549. There were 8 goals on which *slam* was successful, while all variants of *metis* failed with a timeout of 2 seconds.

The results show that *slam*, despite lagging behind in performance on first-order goals, is still able to prove some goals on which *metis* fails because their proof involves

[2] https://www.isa-afp.org/

higher-order reasoning. I believe that narrowing the performance gap on first-order goals will also improve *slam*'s performance on higher-order goals, making it a useful addition to Sledgehammer's suite of proof reconstruction backends.

Acknowledgment. I would like to thank my advisor Jasmin Blanchette for originally suggesting and helping with this work, and Martin Desharnais for his help with the evaluation.

This research has been co-funded by the European Union (ERC, Nekoka, 101083038). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

[1] Bentkamp, A., Blanchette, J., Tourret, S., Vukmirović, P.: Superposition for Higher-Order Logic. Journal of Automated Reasoning **67**(1), 10 (Jan 2023). https://doi.org/10.1007/s10817-022-09649-9, https://doi.org/10.1007/s10817-022-09649-9

[2] Bentkamp, A., Blanchette, J., Tourret, S., Vukmirović, P., Waldmann, U.: Superposition with Lambdas. Journal of Automated Reasoning **65**(7), 893–940 (Oct 2021). https://doi.org/10.1007/s10817-021-09595-y, https://doi.org/10.1007/s10817-021-09595-y

[3] Bhayat, A., Suda, M.: A Higher-Order Vampire (Short Paper). In: Benzmüller, C., Heule, M.J., Schmidt, R.A. (eds.) Automated Reasoning. pp. 75–85. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-63498-7_5

[4] Böhme, S.: Proving Theorems of Higher-Order Logic with SMT Solvers. Ph.D. thesis, Technical University Munich (2012), https://nbn-resolving.org/urn:nbn:de:bvb:91-diss-20120511-1084525-1-4

[5] Cruanes, S.: Extending Superposition with Integer Arithmetic, Structural Induction, and Beyond. phdthesis, École polytechnique (Sep 2015), https://hal.science/tel-01223502

[6] Hurd, J.: First-order proof tactics in higher-order logic theorem provers. In: Archer, M., Di Vito, B., Muñoz, C. (eds.) Design and Application of Strategies/Tactics in Higher Order Logics. pp. 56–68. NASA Technical Reports (2003)

[7] Kovács, L., Voronkov, A.: First-Order Theorem Proving and Vampire. In: Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J.M., Mattern, F., Mitchell, J.C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., Sudan, M., Terzopoulos, D., Tygar, D., Vardi, M.Y., Weikum, G., Sharygina, N., Veith, H. (eds.) Computer Aided Verification, vol. 8044, pp. 1–35. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39799-8_1, http://link.springer.com/10.1007/978-3-642-39799-8_1, series Title: Lecture Notes in Computer Science

[8] Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL: a proof assistant for higher-order logic. No. 2283 in Lecture notes in computer science, Springer, Berlin ; New York (2002)

[9] Paulson, L.C.: A generic tableau prover and its integration with isabelle. J. Univers. Comput. Sci. **5**(3), 73–87 (1999). https://doi.org/10.3217/JUCS-005-03-0073, https://doi.org/10.3217/jucs-005-03-0073

[10] Paulson, L.C., Blanchette, J.C.: Three years of experience with sledgehammer, a practical link between automatic and interactive theorem provers. In: Sutcliffe, G., Schulz, S., Ternovska, E. (eds.) The 8th International Workshop on the Implementation of Logics, IWIL 2010, Yogyakarta, Indonesia, October 9, 2011. EPiC Series in Computing, vol. 2, pp. 1–11. EasyChair (2010). https://doi.org/10.29007/36DT, https://doi.org/10.29007/36dt

[11] Paulson, L.C., Susanto, K.W.: Source-Level Proof Reconstruction for Interactive Theorem Proving. In: Schneider, K., Brandt, J. (eds.) Theorem Proving in Higher Order Logics, vol. 4732, pp. 232–245. Springer Berlin Heidelberg, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74591-4_18, http://link.springer.com/10.1007/978-3-540-74591-4_18, iSSN: 0302-9743, 1611-3349 Series Title: Lecture Notes in Computer Science

[12] Schulz, S.: E – A Brainiac Theorem Prover. AI Commun. **15**(2,3), 111–126 (Aug 2002), https://wwwlehre.dhbw-stuttgart.de/~sschulz/PAPERS/Schulz-AICOM-2002.pdf

[13] Schulz, S.: Fingerprint Indexing for Paramodulation and Rewriting. In: Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J.M., Mattern, F., Mitchell, J.C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., Sudan, M., Terzopoulos, D., Tygar, D., Vardi, M.Y., Weikum, G., Gramlich, B., Miller, D., Sattler, U. (eds.) Automated Reasoning, vol. 7364, pp. 477–483. Springer Berlin Heidelberg, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31365-3_37, http://link.springer.com/10.1007/978-3-642-31365-3_37, series Title: Lecture Notes in Computer Science

[14] Schulz, S.: Simple and Efficient Clause Subsumption with Feature Vector Indexing. In: Bonacina, M.P., Stickel, M.E. (eds.) Automated Reasoning and Mathematics, vol. 7788, pp. 45–67. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36675-8_3, https://wwwlehre.dhbw-stuttgart.de/~sschulz/PAPERS/Schulz-IJCAR-WS-2004.pdf, series Title: Lecture Notes in Computer Science

[15] Vukmirović, P., Bentkamp, A., Nummelin, V.: Efficient Full Higher-Order Unification. Logical Methods in Computer Science **Volume 17, Issue 4**, 6919 (Dec 2021). https://doi.org/10.46298/lmcs-17(4:18)2021, https://lmcs.episciences.org/6919

[16] Vukmirović, P., Blanchette, J., Schulz, S.: Extending a High-Performance Prover to Higher-Order Logic. In: Sankaranarayanan, S., Sharygina, N. (eds.) Tools and Algorithms for the Construction and Analysis of Systems. pp. 111–129. Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-30820-8_10