Stronger Higher-Order Automation: A Report on the Ongoing Matryoshka Project

Jasmin Blanchette[1,2,3], Pascal Fontaine[3], Stephan Schulz[4], Sophie Tourret[2], and Uwe Waldmann[2]

[1] Vrije Universiteit Amsterdam, The Netherlands
j.c.blanchette@vu.nl
[2] Max-Planck-Institut für Informatik, Saarbrücken, Germany
{jasmin.blanchette,sophie.tourret,uwe}@mpi-inf.mpg.de
[3] Université de Lorraine, CNRS, Inria, LORIA, Nancy, France
pascal.fontaine@loria.fr
[4] DHBW Stuttgart, Germany
schulz@eprover.org

Abstract

This extended abstract presents the contributions to automated reasoning made in the context of the project Matryoshka, funded for five years by the European Research Council.

Interactive theorem proving (ITP) is concerned with using proof assistants to write computer-checked proofs of theorems, generally expressed in a variant of higher-order logic (HOL). Proof assistants are expressive and versatile tools, and their success stories range from software and hardware verification [23] to pure mathematics [22]. Despite the higher trustworthiness of a machine-checked proof compared with a hand-written one, only a very small fraction of today's software and mathematical proofs is verified in a proof assistant. One recurrent complaint about these tools is their lack of automation, which has let to the creation of "hammers" [16,26] that attempt to delegate the proof obligations to fully automatic theorem proving (ATP) tools.

Although some automatic theorem provers for HOL exist [14, 33], they tend to underperform on problems generated from proof assistants [34]. Thus, hammers rely mostly on automatic provers for first-order logic (FOL), such as SMT (satisfiability modulo theories) solvers [11, 17] and superposition-based theorem provers [27,32]. One unpleasant property of hammers is that they obfuscate the structure of terms and formulas through the application of a sequence of encodings. This complicates the task of first-order automatic provers since it is this very structure that they exploit to find proofs efficiently.

To some extent, the ITP and ATP research communities have been working in parallel, largely ignoring each other. Hammers have contributed to bringing the two communities closer. Two years ago, a new division was created in the CASC competition, featuring benchmarks generated by Isabelle's Sledgehammer [35], reflecting the growing interest of the ATP community in ITP. Next year, IJCAR will also encompass the ITP conference.

The Matryoshka project, whose general aim is to bridge the gap between ATP and ITP by strengthening higher-order proof automation, is part of this trend. So is our colleagues' work on Leo-III [33] and Vampire [10]. Since Matryoshka's start almost two and a half years ago, progress has been achieved both in terms of improving ATP with an eye on ITP applications and of using ITP as a vehicle for ATP research, taking the form of several workshops and many publications.[1] This year's edition of the ARCADE workshop is a good opportunity to provide to the community a comprehensive view of our contributions and to sketch general avenues for future work.

[1] http://matryoshka.gforge.inria.fr/#Publications

1 Automatic Proof at the Service of Interactive Proof

To face the challenges offered by HOL starting from FOL, we have chosen as milestones the intermediate logics *λ-free HOL* and *Boolean-free HOL*. In λ-free HOL, the only higher-order features allowed are functional variables and partial application of functions. In Boolean-free HOL, λ-expressions are additionally allowed. This fragment is encoded in λ-free HOL by relying on combinators, and λ-free HOL itself is encoded in FOL using the applicative encoding [15]. In these two intermediate logics, variables cannot be of Boolean type. This is the only difference between Boolean-free and full HOL. Both λ-free and Boolean-free HOL are interpreted using Henkin-style semantics [9].

To improve automation in proof assistants, our strategy is to develop calculi for these fragments that extend first-order calculi in a *graceful* way, meaning that the new calculi should be almost as efficient at first-order reasoning as their first-order counterparts while being additionally able to cope with higher-order reasoning steps in a reasonable manner. Our starting points are the superposition calculus and SMT's CDCL(T) calculus, the most successful first-order approaches for solving hammer benchmarks.

Toward Higher-Order Superposition The calculus that Bentkamp, Blanchette, Cruanes, and Waldmann [8] designed for λ-free HOL resembles the first-order superposition calculus. A key ingredient is a term order for λ-free terms. We have developed two such orders that generalize the familiar lexicographic path order (Blanchette, Waldmann, and Wand [13]) and the Knuth–Bendix order with argument coefficients (Becker, Blanchette, Waldmann, and Wand [6]). Both orders are compatible with function contexts, just as their first-order counterparts. In the higher-order case, however, one would also like to have compatibility with arguments (that is, $s \succ s'$ implies $s\,t \succ s'\,t$), but this is hard to obtain. We compensate for this deficiency by restricting the superposition rule to apply at argument subterms (e.g., a or b in f a b, but not f or f a). Other superpositions are made unnecessary by an additional inference rule that adds arguments to partially applied functions.

Adding support for λ-terms imposes major modifications on the calculus (Bentkamp, Blanchette, Tourret, Vukmirović, and Waldmann [7]). Unification is no longer unitary, but infinitary, which means that inference rules generate streams of conclusions for a fixed set of premises. Handling these in practice requires dovetailing. Even worse, the structure of some kinds of variable-headed terms and λ-expressions (called *fluid* terms) can be fundamentally altered under substitution, which breaks the connection between superpositions on clauses and on their ground instances and makes the calculus incomplete. We restore completeness with a new inference rule in which fluid contexts are encoded by a fresh higher-order variable. To reach full HOL, this calculus must be further extended to handle Boolean variables, which can be replaced by arbitrary predicates under substitution, thus breaking the clausal structure of the main formula. Boolean encodings [15] and lazy clausification [21] are promising options to face this final challenge.

A prototype implementation of the λ-free calculus, based on the superposition prover Zipperposition, was implemented by Bentkamp [8]. This prover was then extended to Boolean-free HOL by Bentkamp, Vukmirović, and Tourret with promising results. Following on the same path, Vukmirović, with some guidance from Blanchette, Cruanes, and Schulz, also extended the high-performance E prover to λ-free HOL [36]. Partly by chance, the data structures in E were quite easy to adapt. Major changes include the replacement of the simple sort system by a proper type system. λ-free HOL terms are represented using E's existing term data structures. Types are represented as recursive structures and are maximally shared in a type bank inspired by E's shared term architecture. Updated unification and matching algorithms are able to return partial matches and unifiers. All of E's indexing data structures (perfect discrimination trees [24], fingerprint indices [30], and feature vector indices [31]) have successfully been adapted to λ-free HOL. As a result, the extended E runs at essentially the same speed as the first-order E. The next planned step is the extension of E to Boolean-free HOL.

Toward Higher-Order SMT Together with Barbosa, Barrett, Reynolds, and Tinelli from the CVC4 team, the Matryoshka team, and more specifically El Ouraoui, Fontaine, and Tourret are experimenting with different approaches for extending SMT to HOL. The first, pragmatic approach, targets existing state-of-the-art SMT solvers with large code bases and complex data structures optimized for the first-order case. In this approach, the SMT solver is extended with only minimal modifications to its core data structures and algorithms. The second approach, requiring substantial modifications, entails the redesign of the essential data structures and algorithms of the solver to work directly in λ-free HOL. The approaches have been respectively implemented in the CVC4 and veriT solvers [5].

The pragmatic approach is based on the applicative encoding of HOL [15]. A preprocessing phase eliminates λ-expressions using λ-lifting: Each λ-expression is replaced by a fresh function symbol, and a quantified formula is introduced to define the symbol in terms of the original expression. Then, the applicative encoding is used lazily to cope with partial applications on the ground level. Trigger-based quantifier instantiation is adapted to handle encoded HOL terms properly. A trick that proves useful in practice is to add axioms that help the instantiation technique to go slightly beyond pure λ-free HOL. More precisely, the "store" axiom (stating, for any unary function f, the existence of a function that coincides with f except for one domain value) allows CVC4 to prove quite a few more formulas from the TPTP library when chosen instances are added [5]. In future work, the other instantiation methods, model-based, enumeration-based, and conflict-based, will be similarly adapted.

In the redesign approach, a suboptimal but simple classical congruence closure algorithm is extended to handle ground HOL terms. This algorithm handles partial applications natively, without relying on the applicative encoding. It is, however, necessary to add axioms to support extensionality. The trigger-based quantifier instantiation method is also adapted to cope with λ-free HOL terms. This involves the extension of the indexing techniques and of the matching algorithms. Store axioms are currently not used in the redesign approach, and we believe this partly explains (together with weaker instantiation techniques in general) the gap of efficiency between veriT (implementing the redesign approach) and CVC4 (implementing the pragmatic approach). Our ambition for the near future is to adapt the theory of the Congruence Closure with Free Variables algorithm [4] to λ-free HOL. An implementation of this algorithm would directly allow us to lift conflict-based instantiation to this logic. Redesigning enumerative instantiation techniques is also on our agenda.

Better SMT Proofs for Smooth Reconstruction An important aspect of automated reasoning at the service of interactive proof is that, beyond checking the validity (or unsatisfiability) of a formula, the solver should provide some evidence for the result. We advocate proofs for SMT solvers. In particular, the veriT SMT solver generates fairly comprehensive proofs, and we are working on the proof production capabilities of the solver to ease the burden on tools that would want to replay proofs. Barbosa, Blanchette, and Fontaine [3] recently developed a method to output explicit proofs for many of the preprocessing steps that usually happen inside SMT solvers. These results have been further polished, along with the proof format, mostly by Fleury and Schurr [2, 20]. In Isabelle, they achieve a reconstruction success rate of about 99%. Our goal is to have a reconstruction success rate of 100%, with a solver that behaves gracefully, i.e., that behaves as much as possible in the same way with proof production enabled as when it is disabled.

2 Interactive Proof at the Service of Automatic Proof

The formal verification of ATP leads to a better understanding of its core mechanisms and strengthen its theoretical results. It also facilitates incremental research by allowing researchers to experiment with existing formalisms and easily identify the consequences of their modifications. Sometimes the outcome of verification is a formal framework that can be instantiated by future applications.

A Verified SAT Solver Fleury, with some help from Lammich and supervision from Blanchette and Weidenbach, formalized the conflict-driven clause learning (CDCL) calculus implemented in most modern SAT solvers [12]. Compared with other SAT solver verification works, this work emphasizes the stepwise refinement methodology and the connection between calculi variants described in the literature. It also considers clause forgetting, solver restarts, and incremental solving, which had not been the focus of formalization before.

Based on the CDCL formalization, Fleury implemented an optimized verified SAT solver called IsaSAT [18, 19]. IsaSAT implements the two-watched-literal scheme and other imperative data structures. From a benchmark suite consisting of the preprocessed SAT Competitions problems from 2009 to 2017, IsaSAT solves 801 problems, compared with 368 for `versat` (the second fastest verified SAT solver) and 1388 for MiniSAT (the baseline reference for SAT solving) [18].

A Verified Ordered Resolution Prover Schlichtkrull, Blanchette, Traytel, and Waldmann [29] formalized in the Isabelle/HOL proof assistant [25] a first-order prover based on ordered resolution with literal selection. They followed Bachmair and Ganzinger's account [1] from Chapter 2 of the *Handbook of Automated Reasoning*. The formal development covers the refutational completeness of two resolution calculi for ground clauses and general infrastructure for theorem proving processes and redundancy, culminating with a completeness proof for a first-order prover, called RP, expressed as transition rules operating on triples of clause sets. This material corresponds to the first four sections of Chapter 2. Interestingly, Bachmair and Ganzinger's main completeness result does not hold as stated, due to the improper treatment of inferences involving several copies of the same premise. The formalization uncovered numerous smaller mistakes.

In subsequent work [28], Schlichtkrull, Blanchette, and Traytel specified an executable prover that implements a fixed clause selection strategy and functional data structures, embodying the abstract prover described by Bachmair and Ganzinger. The executable prover is connected to the abstract prover through a chain of refinement steps.

A Framework for Saturation Provers (Ongoing Work) One of the indispensable operations of realistic saturation theorem provers is deletion of subsumed formulas. Unfortunately, the well-known equivalence of dynamic and static refutational completeness holds only for derivations where all deleted formulas are *redundant*, and the usual definition of redundancy does not cover subsumed formulas. The fact that the equivalence of dynamic and static refutational completeness cannot be exploited directly is one of the main reasons why Bachmair and Ganzinger's refutational completeness proof for the RP prover is rather complicated and nonmodular.

Waldmann, Tourret, Robillard, and Blanchette are currently working on a generic framework for formal refutational completeness proofs of abstract provers that implement saturation proof calculi. The framework relies on a modular extension of arbitrary redundancy criteria, which in the end permits not only to cover subsumption deletion, but to model prover architectures in such a way that the static refutational completeness of the calculus immediately implies the dynamic refutational completeness of, say, an Otter loop or Discount loop prover implementing the calculus.

Acknowledgment. We thank Daniel El Ouraoui, Mathias Fleury, Hans-Jörg Schurr, and the anonymous reviewers for suggesting textual improvements. We also thank the ARCADE organizers, reviewers, and attendees for the discussions on this topic. The project receives funding from the European Research Council under the European Union's Horizon 2020 research and innovation program (grant agreement No. 713999, Matryoshka).

References

[1] Leo Bachmair and Harald Ganzinger. Resolution theorem proving. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, volume I, pages 19–99. Elsevier and MIT Press, 2001.

[2] Haniel Barbosa, Jasmin Christian Blanchette, Mathias Fleury, Pascal Fontaine, and Hans-Jörg Schurr. Better SMT proofs for easier reconstruction. In *AITP 2019—Fourth Conference on Artificial Intelligence and Theorem Proving—Abstracts of the Talks—April 7–12, 2019, Obergurgl, Austria*, 2019.

[3] Haniel Barbosa, Jasmin Christian Blanchette, and Pascal Fontaine. Scalable fine-grained proofs for formula processing. In *Automated Deduction—CADE 26—26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6–11, 2017, Proceedings*, volume 10395 of *LNCS*, pages 398–412. Springer, 2017.

[4] Haniel Barbosa, Pascal Fontaine, and Andrew Reynolds. Congruence closure with free variables. In *Tools and Algorithms for the Construction and Analysis of Systems—23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22–29, 2017, Proceedings, Part II*, volume 10206 of *LNCS*, pages 214–230. Springer, 2017.

[5] Haniel Barbosa, Andrew Reynolds, Daniel El Ouraoui, Cesare Tinelli, and Clark Barrett. Extending SMT solvers to higher-order logic. In *Automated Deduction—CADE-27—27th International Conference on Automated Deduction, Natal, Brazil, August 23–30, 2019, Proceedings*, volume 11716 of *LNCS*. Springer, 2019.

[6] Heiko Becker, Jasmin Christian Blanchette, Uwe Waldmann, and Daniel Wand. A transfinite Knuth–Bendix order for lambda-free higher-order terms. In *Automated Deduction—CADE 26—26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6–11, 2017, Proceedings*, volume 10395 of *LNCS*, pages 432–453. Springer, 2017.

[7] Alexander Bentkamp, Jasmin Blanchette, Sophie Tourret, Petar Vukmirović, and Uwe Waldmann. Superposition with lambdas. In *Automated Deduction—CADE-27—27th International Conference on Automated Deduction, Natal, Brazil, August 23–30, 2019, Proceedings*, volume 11716 of *LNCS*. Springer, 2019.

[8] Alexander Bentkamp, Jasmin Christian Blanchette, Simon Cruanes, and Uwe Waldmann. Superposition for lambda-free higher-order logic. In *Automated Reasoning—9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FLoC 2018, Oxford, UK, July 14–17, 2018, Proceedings*, volume 10900 of *LNCS*, pages 28–46. Springer, 2018.

[9] Christoph Benzmüller and Dale Miller. Automation of higher-order logic. In Jörg H. Siekmann, editor, *Computational Logic*, volume 9 of *Handbook of the History of Logic*, pages 215–254. Elsevier, 2014.

[10] Ahmed Bhayat and Giles Reger. Restricted combinatory unification. In *Automated Deduction—CADE-27—27th International Conference on Automated Deduction, Natal, Brazil, August 23–30, 2019, Proceedings*, volume 11716 of *LNCS*. Springer, 2019.

[11] Nikolaj Bjørner. Z3 and SMT in industrial R&D. In *Formal Methods—22nd International Symposium, FM 2018, Held as Part of the Federated Logic Conference, FLoC 2018, Oxford, UK, July 15–17, 2018, Proceedings*, volume 10951 of *LNCS*, pages 675–678. Springer, 2018.

[12] Jasmin Christian Blanchette, Mathias Fleury, Peter Lammich, and Christoph Weidenbach. A verified SAT solver framework with learn, forget, restart, and incrementality. *J. Autom. Reasoning*, 61(1–4):333–365, 2018.

[13] Jasmin Christian Blanchette, Uwe Waldmann, and Daniel Wand. A lambda-free higher-order recursive path order. In *Foundations of Software Science and Computation Structures—20th International Conference, FoSSaCS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22–29, 2017, Proceedings*, volume 10203 of *LNCS*, pages 461–479. Springer, 2017.

[14] Chad E. Brown. Satallax: An automatic higher-order prover. In *Automated Reasoning—6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26–29, 2012. Proceedings*, volume 7364 of *LNCS*, pages 111–117. Springer, 2012.

[15] Haskell Curry, Robert Feys, and William Craig. *Combinatory Logic*, volume I. North-Holland, 1958.

[16] Lukasz Czajka and Cezary Kaliszyk. Hammer for Coq: Automation for dependent type theory. *J. Autom.*

Reasoning, 61(1–4):423–453, 2018.

[17] Morgan Deters, Andrew Reynolds, Tim King, Clark W. Barrett, and Cesare Tinelli. A tour of CVC4: How it works, and how to use it. In *Formal Methods in Computer-Aided Design, FMCAD 2014, Lausanne, Switzerland, October 21–24, 2014*, page 7. IEEE, 2014.

[18] Mathias Fleury. Optimizing a verified SAT solver. In *NASA Formal Methods—11th International Symposium, NFM 2019, Houston, TX, USA, May 7–9, 2019, Proceedings*, volume 11460 of *LNCS*, pages 148–165. Springer, 2019.

[19] Mathias Fleury, Jasmin Christian Blanchette, and Peter Lammich. A verified SAT solver with watched literals using imperative HOL. In *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2018, Los Angeles, CA, USA, January 8–9, 2018*, pages 158–171, 2018.

[20] Mathias Fleury and Hans-Jörg Schurr. Reconstructing veriT proofs in Isabelle/HOL. In *Sixth Workshop on Proof eXchange for Theorem Proving—PxTP 2019—Affiliated with the 27th International Conference on Automated Deduction (CADE-27)—26 August 2019, Natal, Brazil*, 2019.

[21] Harald Ganzinger and Jürgen Stuber. Superposition with equivalence reasoning and delayed clause normal form transformation. In *Automated Deduction—CADE-19—19th International Conference on Automated Deduction Miami Beach, FL, USA, July 28–August 2, 2003, Proceedings*, volume 2741 of *LNCS*, pages 335–349. Springer, 2003.

[22] Thomas Hales, Mark Adams, Gertrud Bauer, Tat Dat Dang, John Harrison, Le Truong Hoang, Cezary Kaliszyk, Victor Magron, Sean McLaughlin, Tat Thang Nguyen, Quang Truong Nguyen, Tobias Nipkow, Steven Obua, Joseph Pleso, Rute Jason, Solovyev Alexey, Thi Hoai An Ta, Nam Trung Tran, Thi Diep Trieu, Josef Urban, Ky Vu, and Roland Zumkeller. A formal proof of the Kepler conjecture. *Forum of Mathematics, Pi*, 5:E2, 2017.

[23] Xavier Leroy. Formal verification of a realistic compiler. *Commun. ACM*, 52(7):107–115, 2009.

[24] William McCune. Experiments with discrimination-tree indexing and path indexing for term retrieval. *J. Autom. Reasoning*, 9(2):147–167, 1992.

[25] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer, 2002.

[26] Lawrence C. Paulson and Jasmin Christian Blanchette. Three years of experience with Sledgehammer, a practical link between automatic and interactive theorem provers. In *8th International Workshop on the Implementation of Logics, IWIL 2010, Yogyakarta, Indonesia, October 9, 2011*, volume 2 of *EPiC Series in Computing*, pages 1–11. EasyChair, 2012.

[27] Giles Reger and Martin Suda. Incremental solving with Vampire. In *Vampire 2017—Proceedings of the 4th Vampire Workshop*, volume 53 of *EPiC Series in Computing*, pages 52–63. EasyChair, 2018.

[28] Anders Schlichtkrull, Jasmin Christian Blanchette, and Dmitriy Traytel. A verified prover based on ordered resolution. In *Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2019, Cascais, Portugal, January 14–15, 2019*, pages 152–165, 2019.

[29] Anders Schlichtkrull, Jasmin Christian Blanchette, Dmitriy Traytel, and Uwe Waldmann. Formalizing Bachmair and Ganzinger's ordered resolution prover. In *Automated Reasoning—9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FLoC 2018, Oxford, UK, July 14–17, 2018, Proceedings*, pages 89–107, 2018.

[30] Stephan Schulz. Fingerprint indexing for paramodulation and rewriting. In *Automated Reasoning—6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings*, volume 7364 of *LNCS*, pages 477–483. Springer, 2012.

[31] Stephan Schulz. Simple and efficient clause subsumption with feature vector indexing. In Maria Paola Bonacina and Mark E. Stickel, editors, *Automated Reasoning and Mathematics: Essays in Memory of William W. McCune*, volume 7788 of *LNCS*, pages 45–67. Springer, 2013.

[32] Stephan Schulz. System description: E 1.8. In *Logic for Programming, Artificial Intelligence, and Reasoning—19th International Conference, LPAR-19, Stellenbosch, South Africa, December 14–19, 2013. Proceedings*, volume 8312 of *LNCS*, pages 735–743. Springer, 2013.

[33] Alexander Steen and Christoph Benzmüller. The higher-order prover Leo-III. In *Automated Reasoning—9th*

International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FLoC 2018, Oxford, UK, July 14–17, 2018, Proceedings, volume 10900 of *LNCS*, pages 108–116. Springer, 2018.

[34] Nik Sultana, Jasmin Christian Blanchette, and Lawrence C. Paulson. LEO-II and Satallax on the Sledgehammer test bench. *J. Applied Logic*, 11(1):91–102, 2013.

[35] Geoff Sutcliffe. The CADE-26 automated theorem proving system competition—CASC-26. *AI Commun.*, 30(6):419–432, 2017.

[36] Petar Vukmirović, Jasmin Christian Blanchette, Simon Cruanes, and Stephan Schulz. Extending a brainiac prover to lambda-free higher-order logic. In *Tools and Algorithms for the Construction and Analysis of Systems—25th International Conference, TACAS 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6–11, 2019, Proceedings, Part I*, volume 11427 of *LNCS*, pages 192–210. Springer, 2019.