


# $\lambda$ -Superposition: From Theory to Trophy

Jasmin Blanchette<sup>1,2,3</sup> 

<sup>1</sup> Ludwig-Maximilians-Universität München, Munich, Germany  
jasmin.blanchette@lmu.de

<sup>2</sup> Max-Planck-Institut für Informatik, Saarland Informatics Campus,  
Saarbrücken, Germany

<sup>3</sup> Université de Lorraine, CNRS, Inria, LORIA, Nancy, France

This extended abstract describes work performed in collaboration with Alexander Bentkamp, Simon Cruanes, Visa Nummelin, Stephan Schulz, Sophie Tourret, Petar Vukmirović, and Uwe Waldmann on the design and implementation of  $\lambda$ -superposition, in the context of the Matroyshka research project.

When I conceived Matroyshka in 2015, my ambition was to develop higher-order provers that perform well on higher-order proof obligations originating from Isabelle/HOL [11] and other proof assistants. Lawrence Paulson had noticed that the performance on truly higher-order goals left much to be desired and “given the inherent difficulty of performing higher-order reasoning using first-order theorem provers, the way forward is to integrate Sledgehammer with an actual higher-order theorem prover, such as LEO-II” [13]. However, the subsequent integration of LEO-II [4] and Satallax [7] failed to bring the expected benefits [16]. My hypothesis was that most Isabelle problems have a large first-order component and the existing higher-order provers were not optimized for this kind of reasoning.

To obtain higher-order provers that excel at first-order reasoning, I proposed to start with a highly successful first-order calculus, superposition, and generalize it, as much as possible, in a graceful way, culminating with a higher-order calculus we call  $\lambda$ -superposition. Provers implementing this calculus would combine the strengths of native higher-order provers and the strengths of the superposition provers that served as Sledgehammer backends: E [14], SPASS [6], and Vampire [5].

To tackle the challenge of designing  $\lambda$ -superposition, we identified three milestones that we reached in turn. We first designed a superposition-like calculus for  $\lambda$ -free, Boolean-free higher-order logic (also called applicative first-order logic) [1]. This logic supports partial application of function symbols (e.g.,  $f$  or  $f\ a$ , where  $f$  is binary) and application of variables (e.g.,  $y\ a$ ). Already at this stage, the first serious issue arose with the term order that superposition uses to prune the search space. We were able to work around the issue by introducing a new inference rule called argument congruence. For this and the other milestones, much of the work went into ensuring that the calculus is refutationally complete.

For the second milestone, we designed a superposition-like calculus for a logic that supports  $\lambda$ -abstractions but not interpreted Booleans [3]. One difficulty that arose is that inferences need to perform higher-order unification. Unfortunately, higher-order unification is ill-behaved: It is undecidable and can yield a possibly infinite stream of unifiers. Moreover, due to interactions with the term order, we need to perform full unification (including flex-flex pairs) [17] and not simply preunification [10].

For the third milestone, we added support for interpreted Booleans [2]. This step was based on ideas by Ganzinger and Stuber [9]. They showed how to support logical symbols inside a superposition-like calculus, but fell short of including an interpreted Boolean type. Thus, we extended Ganzinger and Stuber’s work [12] and used it as the basis of a graceful generalization to higher-order logic.

Whenever we designed a calculus, we also made sure to implement it in the Zipperposition prover [8]. Zipperposition was originally developed by Cruanes to explore induction, arithmetic, and deduction modulo. It is written in OCaml and is highly extensible. He extended it with a pragmatic higher-order mode with support for  $\lambda$ -abstractions and extensionality, without any completeness guarantees. This mode formed the basis for our subsequent work. Empirical evaluations on TPTP and Sledgehammer benchmarks were initially disappointing, but after some extensive tuning and new ideas for heuristics, Zipperposition became highly competitive, finishing first in the higher-order theorem division of the CADE ATP System Competition (CASC) in 2020, 2021, and 2022. Inspired by a similar integration in Leo-III [15] and Satallax, Zipperposition incorporates E as a backend to tackle first-order subproblems.

We also implemented  $\lambda$ -superposition in the high-performance prover E [18,19]. The E implementation is pragmatic and sacrifices completeness. For example, the possibly infinite stream of unifiers is truncated to make it finite, and some of the most explosive rules of  $\lambda$ -superposition are omitted. Probably because Zipperposition has a portfolio of modes extensively tuned against the TPTP library and uses a version of E as a backend, E finished only second in the higher-order theorem division of CASC 2022. On the other hand, E finished first in the Sledgehammer division of the same competition. Despite this, the performance improvement over Sledgehammer’s first-order backends is unimpressive. I suspect that Isabelle problems are even more first-order than I thought.

We learned a few other lessons in the process:

- The identification of reasonable milestones was invaluable.
- The completeness proofs gave us some useful guidance as we designed the calculi, even if it turns out that the best empirical modes are incomplete.
- Another useful guide was the design goal of achieving, as much as possible, a graceful generalization, preserving the features that make standard superposition successful on first-order problems.
- Disappointing evaluations can simply mean that more fine-tuning and heuristics are needed.
- The presence of many complementary modes in a well-tuned portfolio can be as important as a highly efficient implementation.

**Acknowledgment.** I thank Alexander Bentkamp, Stephan Schulz, Mark Summerfield, Petar Vukmirović, and Uwe Waldmann for textual suggestions. This research has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation program (grant No. 713999, Matryoshka). This research has also received funding from the Netherlands Organization for Scientific Research (NWO) under the Vidi program (project No. 016.Vidi.189.037, Lean Forward).

## References

- [1] Bentkamp, A., Blanchette, J., Cruanes, S., Waldmann, U.: Superposition for lambda-free higher-order logic. *Log. Meth. Comput. Sci.* 17(2), 1:1–1:38 (2021)
- [2] Bentkamp, A., Blanchette, J., Tournet, S., Vukmirović, P.: Superposition for higher-order logic. *J. Autom. Reason.* 67(1), article 10 (2023)
- [3] Bentkamp, A., Blanchette, J., Tournet, S., Vukmirović, P., Waldmann, U.: Superposition with lambdas. *J. Autom. Reason.* 65(7), 893–940 (2021)
- [4] Benz Müller, C., Paulson, L.C., Theiss, F., Fietzke, A.: LEO-II—A cooperative automatic theorem prover for higher-order logic. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) *IJCAR 2008*. LNCS, vol. 5195, pp. 162–170. Springer (2008)
- [5] Bhayat, A., Rege, G.: A combinator-based superposition calculus for higher-order logic. In: Peltier, N., Sofronie-Stokkermans, V. (eds.) *IJCAR 2020, Part I*. LNCS, vol. 12166, pp. 278–296. Springer (2020)
- [6] Blanchette, J.C., Popescu, A., Wand, D., Weidenbach, C.: More SPASS with Isabelle: Superposition with hard sorts and configurable simplification. In: Beringer, L., Felty, A. (eds.) *ITP 2012*. LNCS, vol. 7406, pp. 345–360. Springer (2012)
- [7] Brown, C.E.: Satallax: An automatic higher-order prover. In: Gramlich, B., Miller, D., Sattler, U. (eds.) *IJCAR 2012*. LNCS, vol. 7364, pp. 111–117. Springer (2012)
- [8] Cruanes, S.: Extending Superposition with Integer Arithmetic, Structural Induction, and Beyond. PhD thesis, École polytechnique (2015)
- [9] Ganzinger, H., Stuber, J.: Superposition with equivalence reasoning and delayed clause normal form transformation. *Inform. Comput.* 199(1–2), 3–23 (2005)
- [10] Huet, G.P.: A unification algorithm for typed lambda-calculus. *Theor. Comput. Sci.* 1(1), 27–57 (1975)
- [11] Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL: A Proof Assistant for Higher-Order Logic, LNCS, vol. 2283. Springer (2002)
- [12] Nummelin, V., Bentkamp, A., Tournet, S., Vukmirović, P.: Superposition with first-class Booleans and inprocessing clausification. In: Platzer, A., Sutcliffe, G. (eds.) *CADE-28*. LNCS, Springer (2021)
- [13] Paulson, L.C.: Three years of experience with Sledgehammer, a practical link between automated and interactive theorem provers. In: Konev, B., Schmidt, R., Schulz, S. (eds.) *PAAR-2010* (2010)
- [14] Schulz, S., Cruanes, S., Vukmirović, P.: Faster, higher, stronger: E 2.3. In: Fontaine, P. (ed.) *CADE-27*. LNCS, vol. 11716, pp. 495–507. Springer (2019)
- [15] Steen, A., Benz Müller, C.: The higher-order prover Leo-III. In: Galmiche, D., Schulz, S., Sebastiani, R. (eds.) *IJCAR 2018*. LNCS, vol. 10900, pp. 108–116. Springer (2018)
- [16] Sultana, N., Blanchette, J.C., Paulson, L.C.: LEO-II and Satallax on the Sledgehammer test bench. *J. Appl. Log.* 11(1), 91–102 (2013)
- [17] Vukmirović, P., Bentkamp, A., Nummelin, V.: Efficient full higher-order unification. In: Ariola, Z.M. (ed.) *FSCD 2020*. LIPIcs, vol. 167, pp. 5:1–5:17. Schloss Dagstuhl—Leibniz-Zentrum für Informatik (2020)
- [18] Vukmirović, P., Blanchette, J., Cruanes, S., Schulz, S.: Extending a brainiac prover to lambda-free higher-order logic. *Int. J. Softw. Tools Technol. Transf.* 24(1), 67–87 (2022)
- [19] Vukmirović, P., Blanchette, J., Schulz, S.: Extending a high-performance prover to higher-order logic. In: Sharygina, N., Sankaranarayanan, S. (eds.) *TACAS 2023*. LNCS, Springer (2023)