Mechanized metatheory revisited (abstract)

Dale Miller

Inria & LIX/École polytechnique
dale.miller@inria.fr

Over a decade ago, the POPLmark challenge [2] suggested that the theorem proving community had tools that were close to being usable by programming language researchers to formally prove properties of their designs and implementations. The authors of the POPLmark challenge looked at existing practices and systems and urged the developers of proof assistants to make improvements to existing systems.

> Our conclusion from these experiments is that the relevant technology has developed almost to the point where it can be widely used by language researchers. We seek to push it over the threshold, making the use of proof tools common practice in programming language research—mechanized metatheory for the masses. [2]

In fact, a number of research teams have used proof assistants to formally proved significant properties of entire programming languages. Such properties include type preservation, determinancy of evaluation, and the correctness of an OS microkernel and of various compilers: see, for example, [9, 10, 11, 15].

As noted in [2], the poor support for binders in syntax was one problem that held back proof assistants from achieving even more widespread use by programming language researchers and practitioners. In recent years, a number of extensions to programming languages and to proof assistants have been developed for treating bindings. These go by names such as locally nameless [4, 18], nominal reasoning [1, 5, 17, 19], and parametric higher-order abstract syntax [6]. Some of these approaches involve extending underlying programming language implementations while the others do not extend the proof assistant or programming language but provide packages, libraries, and/or abstract datatypes that attempt to hide and orchestrate various issues surrounding the syntax of bindings. In the end, nothing canonical seems to have appeared since the POPLmark challenge was made: we are left with a simple grid that rates different approaches on various attributes [16].

Clearly, mature and extensible proof assistants, such as, say, Coq, HOL, and Isabelle/HOL can be extended to deal with syntactic challenges (such as bindings in syntax) that they were not originally designed to handle. At the same time, it seems plausible and desirable to pursue approaches to the problem of bindings in syntax and metatheory more generally.

In this talk, I will argue that bindings are such an intimate aspect of the structure of expressions that they should be accounted for directly in the underlying programming language support for proof assistants. High-level and semantically elegant programming language support can be found in rather old and familiar concepts. In particular, Church's Simple Theory of Types [7] has long ago provided answers to how bindings interact with logical connectives and quantifiers. Similarly, the proof search interpretation [13] of Gentzen's proof theory [8] provides a rich model of computation that supports bindings. I outline several principles for dealing computationally with bindings that follow from their treatments in quantificational logic and sequent calculus. One of the most central principles about bindings is that bound variables never become free: instead bindings can move from term-level bindings (λ-abstractions) to formula-level bindings (quantifiers) to proof-level bindings (eigenvariables and nominal constants) [12, 14] I will also describe some implementations [3, 12] of these principles that have helped to validate their effectiveness as computational principles.

Acknowledgments. This work has been funded by the ERC Advanced Grant ProofCert.

References

[1] Brian Aydemir, Aaron Bohannon, and Stephanie Weirich. Nominal reasoning techniques in Coq. In *International Workshop on Logical Frameworks and Meta-Languages:Theory and Practice (LFMTP)*, pages 69–77, Seattle, WA, USA, August 2006.

[2] Brian E. Aydemir, Aaron Bohannon, Matthew Fairbairn, J. Nathan Foster, Benjamin C. Pierce, Peter Sewell, Dimitrios Vytiniotis, Geoffrey Washburn, Stephanie Weirich, and Steve Zdancewic. Mechanized metatheory for the masses: The POPLmark challenge. In *Theorem Proving in Higher Order Logics: 18th International Conference*, number 3603 in LNCS, pages 50–65. Springer, 2005.

[3] David Baelde, Kaustuv Chaudhuri, Andrew Gacek, Dale Miller, Gopalan Nadathur, Alwen Tiu, and Yuting Wang. Abella: A system for reasoning about relational specifications. *Journal of Formalized Reasoning*, 7(2), 2014.

[4] Arthur Charguéraud. The locally nameless representation. *Journal of Automated Reasoning*, pages 1–46, May 2011.

[5] James Cheney and Christian Urban. Nominal logic programming. *ACM Trans. Program. Lang. Syst.*, 30(5):1–47, 2008.

[6] Adam Chlipala. Parametric higher-order abstract syntax for mechanized semantics. In James Hook and Peter Thiemann, editors, *Proceeding of the 13th ACM SIGPLAN international conference on Functional programming, ICFP 2008, Victoria, BC, Canada, September 20-28, 2008*, pages 143–156. ACM, 2008.

[7] Alonzo Church. A formulation of the Simple Theory of Types. *J. of Symbolic Logic*, 5:56–68, 1940.

[8] Gerhard Gentzen. Investigations into logical deduction. In M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, pages 68–131. North-Holland, Amsterdam, 1935.

[9] Gerwin Klein, Kevin Elphinstone, Gernot Heiser, June Andronick, David Cock, Philip Derrin, Dhammika Elkaduwe, Kai Engelhardt, Rafal Kolanski, Michael Norrish, Thomas Sewell, Harvey Tuch, and Simon Winwood. seL4: Formal verification of an OS kernel. In *Proceedings of the 22nd Symposium on Operating Systems Principles (22nd SOSP'09), Operating Systems Review (OSR)*, pages 207–220, Big Sky, MT, October 2009. ACM SIGOPS.

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

[11] Donald MacKenzie. *Mechanizing Proof*. MIT Press, 2001.

[12] Dale Miller and Gopalan Nadathur. *Programming with Higher-Order Logic*. Cambridge University Press, June 2012.

[13] Dale Miller, Gopalan Nadathur, Frank Pfenning, and Andre Scedrov. Uniform proofs as a foundation for logic programming. *Annals of Pure and Applied Logic*, 51:125–157, 1991.

[14] Dale Miller and Alwen Tiu. A proof theory for generic judgments. *ACM Trans. on Computational Logic*, 6(4):749–783, October 2005.

[15] J. Strother Moore. A mechanically verified language implementation. *J. of Automated Reasoning*, 5(4):461–492, 1989.

[16] The POPLmark Challenge webpage. `http://www.seas.upenn.edu/~plclub/poplmark/`, 2015.

[17] François Pottier. An overview of Cαml. In *ACM Workshop on ML*, ENTCS, pages 27–51, September 2005.

[18] Peter Sewell, Francesco Zappa Nardelli, Scott Owens, Gilles Peskine, Thomas Ridge, Susmit Sarkar, and Rok Strniša. Ott: Effective tool support for the working semanticist. *Journal of Functional Programming*, 20(01):71–122, 2010.

[19] Christian Urban. Nominal reasoning techniques in Isabelle/HOL. *Journal of Automated Reasoning*, 40(4):327–356, 2008.