Try hard! Proof automation, efficient-proof-script reconstruction, and proof-planning in Isabelle using monads.

Yutaka Nagashima*
Software Systems Research Group at NICTA

The reliability of critical software systems is essential; even though people's lives are becoming more and more dependent on software systems, a number of security-critical software faults are being discovered.

Type safe languages and static analysis tools are employed to eliminate bugs statically and automatically. However, most of these techniques are inadequate to assure that implementations are functionally correct in terms of their specifications. They only assure the lack of certain bugs, but they do not guarantee *functional correctness*.

There is only one known way to guarantee functional correctness of software artefacts rigorously; specify the desired behaviour formally and mechanically, and prove that the implementation respects the specification using a theorem prover such as Isabelle, Coq, or HOL4.

This approach is known as *complete formal verification*. The seL4 micro-kernel, the C compiler CompCert, and the ML compiler CakeML are major achievements in this field; they are proved to be correct mechanically. These projects have demonstrated that complete formal verification is possible and, to some extent, cost-effective. However, complete formal verification of small software project still requires significant effort.

Many proof automation tools have been developed to alleviate the cost of verification. Most interactive theorem provers come with default proof-procedures called *tactic*s. Despite the support of these tools, interactive theorem proving is still considered as a hard problem requiring experienced engineers.

We believe that there is still significant room to improve proof automation in interactive theorem provers. In this talk, we present our new proof automation tools for Isabelle which produce efficient proof-scripts if they successfully discharge given proof-obligations. Our automation tools are fully embedded in Isabelle, enabling the seamless invocation from Isabelle/jEdit, the standard proof-editor of Isabelle.

Furthermore, we present our *framework* for proof automation. This framework allows Isabelle users to specify their own proof-search procedures *succinctly* as data structures.

"*Succinctly*" is the keyword here; our monadic interpretation of tactics makes our framework compositional and abstract the core of framework away from the implementation details. Isabelle's source language, ML, does not support type classes natively. Therefore, we emulate type classes to implement monads using the ML module system.

*NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council.

This ongoing work has already demonstrated that our approach successfully discharges a set of small-scale proof-obligations that could not be proved by the existing proof-automation tools in Isabelle. Finally, we explain our plan to improve our automation tools and framework, so that our approach can solve larger-scale proof-obligations.

Keywords: interactive theorem proving, proof automation, proof-reconstruction, monadic-interpretation, tactic-language

References

DIXON, L., AND FLEURIOT, J. D. 2003. Isaplanner: A prototype proof planner in isabelle. In *Automated Deduction - CADE-19, 19th International Conference on Automated Deduction Miami Beach, FL, USA, July 28 - August 2, 2003, Proceedings*, 279–283.

DREYER, D., HARPER, R., CHAKRAVARTY, M. M. T., AND KELLER, G. 2007. Modular type classes. In *Proceedings of the 34th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2007, Nice, France, January 17-19, 2007*, 63–70.

KLEIN, G., ANDRONICK, J., ELPHINSTONE, K., HEISER, G., COCK, D., DERRIN, P., ELKADUWE, D., ENGELHARDT, K., KOLANSKI, R., NORRISH, M., SEWELL, T., TUCH, H., AND WINWOOD, S. 2010. sel4: formal verification of an operating-system kernel. *Commun. ACM 53*, 6, 107–115.

KUMAR, R., MYREEN, M. O., NORRISH, M., AND OWENS, S. 2014. Cakeml: a verified implementation of ML. In *The 41st Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL '14, San Diego, CA, USA, January 20-21, 2014*, 179–192.

LEROY, X. 2009. A formally verified compiler back-end. *Journal of Automated Reasoning 43*, 4, 363–446.

MARTIN, A., AND GIBBONS, J., 2002. A monadic interpretation of tactics.

MARTIN, A. P., GARDINER, P. H. B., AND WOODCOCK, J. 1996. A tactic calculus-abridged version. *Formal Asp. Comput. 8*, 4, 479–489.

PAULSON, L. C. 1994. *Isabelle - A Generic Theorem Prover (with a contribution by T. Nipkow)*, vol. 828 of *Lecture Notes in Computer Science*. Springer.

SLIND, K., AND NORRISH, M. 2008. A brief overview of HOL4. In *Theorem Proving in Higher Order Logics, 21st International Conference, TPHOLs 2008, Montreal, Canada, August 18-21, 2008. Proceedings*, 28–32.