Automation by Analogy, in Coq

Alasdair Hill[1]*and Ekaterina Komendantskaya[2]

[1] Heriot-Watt University Edinburgh, UK
ath7@hw.ac.uk
[2] Heriot-Watt University Edinburgh, UK
ek19@hw.ac.uk

Abstract

Automation of popular interactive theorem provers like Coq has become a hot topic, due to the growing number and size of verification projects such provers accommodate. Several automation tools for Coq have been suggested, from advanced tactics like *Crush*, to SMT-solving solutions like *SMT-Coq* and *Hammer for Coq*. In this talk, we will present a complementary set of automation methods for Coq, based on discovery of proof similarities and common proof patterns in the code.

Extended Abstract

The problem of automating (or aiding) Coq proof construction is as old as Coq itself. *Ltac* tactic language is by a wide margin the most popular Coq automation tool to date. Over the years it hosted a range of impressive extensions like e.g. SSReflect [6] and Crush [3]. Neither of these extensions is "AI based", i.e. neither uses automated reasoning or machine learning. Seeing the succes of Isabelle/HOL in AI based automation [2], it is not unreasonable to predict that incorportaion of some kinds of AI based tools in Coq may aid to further automate some aspects of proof development. The main two questions are: (1) what kinds of AI tools? and (2) which aspects of proof development? Very often, the answer to question (1) determines the answer to question (2).

For example, one answer to question (1) is to encorporate powerful SMT solvers into Coq, thus aiming for automation of Coq proofs that correspond to the first-order theory of the underlying SMT solver. For the phase of translation of proofs from the language of the SMT-solver back to Coq, two engineering solutions are possible. Hammers for Coq [4] approach suggests to use the "Hammer" methods [2] also employed in Isabelle and HOL, i.e. to reconstruct the Ltac tactics from SMT proof traces. SMT-Coq [5] approach uses the small scale reflection to *reflect* the proofs generated by the SMT-solver back into Coq's language.

In this talk, we will propose an alternative answer to the questions (1) and (2). We propose to use a method of statistical pattern-recognition to detect structural similarities among Coq proofs and definitions. It has been implemented in ML4PG (*Machine-Learning for Proof General*) [10, 8]. ML4PG performs a structural analysis of all Coq objects in the given libraries, and discovers their mutual dependencies and similarities. Based on the discovered patterns, it outputs small sets (clusters) of similar proofs.

If a theorem of interest belongs to a certain cluster, other lemmas and theorems in that cluster are deemed to be structurally similar to it, and we can try to reconstruct an Ltac proof script for a new theorem by analogy with the Ltac scripts of similar proofs in the cluster. Unlike the SMT-based tools, this method will not be restricted to first-order fragment of Coq proofs, and it will work similarly for SSReflect or plain Coq proofs. But this method will be limited by

*A.Hill is funded by an EPSRC DTA grant.

the power of the **analogical argument**. For example, a new theorem may not be provable by analogy with any other existing theorem, or the analogy may run deeper than any Ltac tactic combination we may generate. The recent preliminary results [11] showed a big variation in success of the analogical method, depending on the libraries, ranging from 94% in HoTT Path library [1], and dropping to 36% in the standard SSReflect library.

In this talk, we will give a detailed experimental study of the power and limitations of the analogical proof reconstruction in Coq setting. We will show four new prototype tools that explore the analogies arising from structural similarities of proofs in four different ways, some involving heuristics such as the automata generation of SEPIA [7]. We compare the performance of these four new analogical methods on SSReflect, CompCert [12], and CoqHoTT libraries.

A similar study of proof automation by analogy has been done in ACL2 [9].

References

[1] S. Awodey, T. Coquand, V. Voevodsky, et al. *Homotopy Type Theory: Univalent Foundations of Mathematics*. http://homotopytypetheory.org/book, Institute for Advanced Study, 2013. https://github.com/HoTT/HoTT/wiki.

[2] Jasmin Christian Blanchette, Cezary Kaliszyk, Lawrence C. Paulson, and Josef Urban. Hammering towards QED. *J. Formalized Reasoning*, 9(1):101–148, 2016.

[3] Adam Chlipala. *Certified Programming with Dependent Types*. MIT Press, 2011.

[4] Lukasz Czajka and Cezary Kaliszyk. Goal translation for a hammer for coq (extended abstract). In *Proceedings First International Workshop on Hammers for Type Theories, HaTT@IJCAR 2016, Coimbra, Portugal, July 1, 2016.*, volume 210 of *EPTCS*, pages 13–20, 2016.

[5] Burak Ekici, Alain Mebsout, Cesare Tinelli, Chantal Keller, Guy Katz, Andrew Reynolds, and Clark W. Barrett. Smtcoq: A plug-in for integrating SMT solvers into coq. In *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part II*, volume 10427 of *Lecture Notes in Computer Science*, pages 126–133. Springer, 2017.

[6] G. Gonthier and A. Mahboubi. An introduction to small scale reflection. *Journal of Formalized Reasoning*, 3(2):95–152, 2010.

[7] Thomas Gransden, Neil Walkinshaw, and Rajeev Raman. SEPIA: search for proofs using inferred automata. In *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, volume 9195 of *Lecture Notes in Computer Science*, pages 246–255. Springer, 2015.

[8] J. Heras and E. Komendantskaya. Recycling Proof Patterns in Coq: Case Studies. *Journal Mathematics in Computer Science*, 2014.

[9] Jónathan Heras, Ekaterina Komendantskaya, Moa Johansson, and Ewen Maclean. Proof-pattern recognition and lemma discovery in ACL2. In *Logic for Programming, Artificial Intelligence, and Reasoning - 19th International Conference, LPAR-19, Stellenbosch, South Africa, December 14-19, 2013. Proceedings*, volume 8312 of *Lecture Notes in Computer Science*, pages 389–406. Springer, 2013.

[10] E. Komendantskaya et al. Machine Learning for Proof General: interfacing interfaces. *Electronic Proceedings in Theoretical Computer Science*, 118:15–41, 2013.

[11] Ekaterina Komendantskaya and Jónathan Heras. Proof mining with dependent types. In *Intelligent Computer Mathematics - 10th International Conference, CICM 2017, Edinburgh, UK, July 17-21, 2017, Proceedings*, volume 10383 of *Lecture Notes in Computer Science*, pages 303–318. Springer, 2017.

[12] X. Leroy. Formal verification of a realistic compiler. *Communications of the ACM*, 52(7):107–115, 2009.