Towards Readable Program Correctness Proofs in Coq[*]

Jacek Chrząszcz and Aleksy Schubert[1]

University of Warsaw, [alx,chrzaszc]@mimuw.edu.pl

Abstract

Coq is a proof assistant that can be used in the process of program verification. We present a number of difficulties one can meet when Coq is used for this purpose and some techniques that can be employed to make the process of proving more comprehensive and accessible to formal proof developers.

Formal verification tools rely on providing detailed account of programs that are verified. All the details there must be mastered explicitly — the tools watch that no detail is overlooked in the implementation. The advantage of this approach is that they can be exploited in scenarios in which programmers should demonstrate their full understanding of code. The disadvantage is that the semantics of the program must be expressed in the native formalism of the tool and the translation itself is complicated. It is so complicated that the usual result is that the final description hardly resembles the original code and a considerable expertise is necessary to understand which particular nuance of the original code is currently being proved.

Even though the progress in automated theorem proving is significant there are and *will be* cases when one has to rely on interactive provers such as Coq to obtain verified piece of code. Actually, the interactive provers are inevitable when there is a mismatch between specification and code. They are difficult to replace when systematic search of mismatch is attempted. We propose to actively address the issues associated with program verification through development of techniques that help proof developers to construct their proofs in interactive environments. To make the process more concrete we demonstrate the usefulness of these techniques in the context of two program verification platforms, one of them being Frama-C coupled with Why3 and the other one HAHA (available from `http://haha.mimuw.edu.pl/`).

Frama-C is a set of tools that can be used to analyse correctness of programs written in the C programming language [1]. It generates Hoare-logic style verification conditions that can be given as input to Why3 and then one can manage their verification in Why3. In particular some of the verification conditions can be proven correct in Coq [2]. One can argue that verification conditions that are obtained from this tool-chain are real-world verification conditions that one should deal with in the process of real program verification.

HAHA is a tool that can be used to teach Hoare-logic basics through verification of programs written in a small imperative programming language. It is developed in Warsaw and there are initial results that indicate that its use can give advantage to students who have contact with it [3]. One of its features is the ability to generate verification conditions provable in Coq. The verification conditions obtained from this tool are simpler than those obtained from Frama-C. However, this has the advantage that they can serve better to illustrate solutions to problems encountered during verification.

In our talk we will compare the two environments and show most problematic situations that emerge during proof development. We divide the issues encountered into two basic categories. One is associated with structuring and presentation of the statements, the second is associated

[*]This work was partially supported by the Polish NCN grant no 2013/11/B/ST6/01381.

with proving mechanisms specific for program verification that can be made available in both of the environments to make proving efforts easier. In particular, the typical situation that needs to be handled is that due to some assignment a small portion of memory is changed, but some property must be carried over through the assignment. We developed a Coq tactic that is able to deal with such situation in many typical situations. We first did it for Coq scripts obtained from HAHA and then adapted to proofs done under Frama-C–Why tool-chain.

References

[1] Pascal Cuoq, Florent Kirchner, Nikolai Kosmatov, Virgile Prevosto, Julien Signoles, and Boris Yakobowski. Frama-C - A software analysis perspective. In George Eleftherakis, Mike Hinchey, and Mike Holcombe, editors, *Proc. of SEFM'12*, volume 7504 of *LNCS*. Springer, 2012.

[2] Jean-Christophe Filliâtre and Andrei Paskevich. Why3 — where programs meet provers. In Matthias Felleisen and Philippa Gardner, editors, *Proc. of ESOP'13*, volume 7792 of *LNCS*, pages 125–128. Springer, 2013.

[3] Tadeusz Sznuk and Aleksy Schubert. Tool support for teaching hoare logic. In Dimitra Giannakopoulou and Gwen Salaün, editors, *Software Engineering and Formal Methods - 12th International Conference, SEFM 2014, Grenoble, France, September 1-5, 2014. Proceedings*, volume 8702 of *LNCS*, pages 332–346, 2014.