Compositional and Customizable Reflective Proofs

Gregory M. Malecha
gmalecha@cs.harvard.edu
Harvard SEAS

Recent years have seen a proliferation of logics, from simple type systems to higher-order program logics dealing with all the intricacies of machine code or concurrency. While developing new logics requires deep technical insight, evaluating their usefulness on real-world programs is a much more Herculean task, since existing automation techniques often do not support the rich features of new logics. In this work we detail a series of technical insights that enable a framework for formal *proof engineering* that supports modularity and extension.

Our approach embraces computational reflection [1], the idea of proving by appeal to verified decision procedures. We extend the proof by reflection approach in three novel ways. First, we support open-ended verification procedures which allows us to carve off different logical domains and reason about them independently while retaining the ability to compose provers over different domains. Second, we support verified user extensions ranging in complexity from simple hints to full-fledged heuristic decision procedures. Dependent types play a crucial role in this, enabling us to package these extensions with their soundness proofs providing a clean verification interface. Third, our decision procedures support binders and unification variables. Combined, these features allow us to both introduce new unification variables and to instantiate existing ones in the style of Coq's "e" tactics such as `eapply` and `eexists`.

In the presentation I describe how the synergy of these techniques can be applied to develop a Coq framework for large-scale reflective decision procedures called MirrorShard [2]. MirrorShard's original goals focus on the the automation of low-level imperative programs verified in higher-order logic and it includes reflective procedures for symbolic execution and entailment checking in separation logic. The current research direction focuses on generalizing these ideas to support more compositionality including the difficulties of dealing with type functions.

References

[1] Samuel Boutin. Using reflection to build efficient and certified decision procedures. In *Proc. TACS*, 1997.

[2] G. Malecha, A.Chlipala, T. Braibant, P. Hulin, and E. Yang. MirrorShard: Proof by Computational Reflection with Verified Hints. *CoRR*, abs/1305.6543, 2013.