Verifying Whiley Programs with Boogie

Mark Utting
Senior Lecturer, ICT
University of the Sunshine Coast
Email: utting@usc.edu.au

Whiley is a verification-friendly programming language that employs extended static checking to eliminate many errors at compile time. It is a hybrid object-oriented and functional programming language, is intended as a platform for research in software verification [2], and comes with its own verification engine for checking program correctness.

This paper describes a translator (Wy2B) from Whiley to Boogie [1]. Boogie is an intermediate verification language from Microsoft Research that is intended as a back-end for multiple kinds of languages The Wy2B translator is being developed as part of a project to develop a weakest precondition semantics for Whiley. The translation of Whiley to Boogie helps to clarify the weakest precondition semantics of Whiley and also gives an alternative verification path for Whiley programs.

We describe several challenges of mapping Whiley to Boogie, such as:

- modelling the rich Whiley type system (structural types, user-defined subtypes, and union, intersection and negation types);
- translating the flow sensitive typing of Whiley, into Boogie's static type system;
- translating type-sensitive equality;
- translating Whiley's *implicit pre/postconditions*, which assume a three-valued logic, into Boogie's two-valued pre/postconditions.

We also discuss the effectiveness of the translator as an alternative verification path, in terms of what Whiley language features can be translated, what percentage of valid Whiley programs can be verified using the Wy2B+Boogie+Z3 toolchain, and the outstanding limitations and challenges.

References

[1] Mike Barnett et al. Boogie: A modular reusable verifier for object-oriented programs. In *Proceedings of the 4th International Conference on Formal Methods for Components and Objects*, FMCO'05, pages 364–387, Berlin, Heidelberg, 2006. Springer-Verlag.

[2] David J. Pearce and Lindsay Groves. Whiley: a platform for research in software verification. In *Proceedings of the Conference on Software Language Engineering (SLE)*, pages 238–248, 2013.