Summary: The quest to develop increasingly sophisticated verification systems continues unabated. Tools such as Dafny, Spec#, ESC/Java, SPARK Ada and Whiley attempt to seamlessly integrate specification and verification into a programming language, in a similar way to type checking. A common integration approach is to generate verification conditions that are handed off to an automated theorem prover. This provides a nice separation of concerns and allows different theorem provers to be used interchangeably. However, generating verification conditions is still a difficult undertaking and the use of more “high-level” intermediate verification languages has become commonplace. In particular, Boogie provides a widely used and understood intermediate verification language. A common difficulty is the potential for an impedance mismatch between the source language and the intermediate verification language. In this paper, we explore the use of Boogie as an intermediate verification language for verifying programs in Whiley. This is noteworthy because the Whiley language has (amongst other things) a rich type system with considerable potential for an impedance mismatch. We provide a comprehensive account of translating Whiley to Boogie which demonstrates that it is possible to model most aspects of the Whiley language. Key challenges posed by the Whiley language included: the encoding of Whiley’s expressive type system and support for flow typing and generics; the implicit assumption that expressions in specifications are well defined; the ability to invoke methods from within expressions; the ability to return multiple values from a function or method; the presence of unrestricted lambda functions; and the limited syntax for framing. We demonstrate that the resulting verification tool can verify significantly more programs than the native Whiley verifier which was custom-built for Whiley verification. Furthermore, our work provides evidence that Boogie is (for the most part) sufficiently general to act as an intermediate language for a wide range of source languages.

MSC:
68N30 Mathematical aspects of software engineering (specification, verification, metrics, requirements, etc.)
68Q60 Specification and verification (program logics, model checking, etc.)
68V15 Theorem proving (automated and interactive theorem provers, deduction, resolution, etc.)

Keywords:
Whiley; Boogie; verifying compiler; intermediate verification language; semantic translation; impedance mismatch; flow typing; verification conditions

Software:
GPUVerify; CVC4; VeriCool; RustHorn; KeY; Dafny; ACSL; Korat; Jahob; Frama-C; SIMPLIFY; Nagini; Whiley; Whiteoak; JMLAutoTest; z3; Crust; Viper; CBMC; Why3; ESC/Java; VCC; Boogie; AutoProof; JMLUnit; OpenJML; Alt-Ergo; SMACK; Joogie; SMT-LIB; Yices; Eiffel; VerCors; Spec; BVD; RustBelt; cvc3

doi:10.1007/978-3-319-49812-6

Full Text: [DOI](https://doi.org/10.1007/978-3-319-49812-6)


[Leino, K.R.M., Müller, P.; Using the Spec# language, methodology, and tools to write bug-free programs. In: LASER Summer School, LNCS, vol. 6029, pp. 91-139. Springer-Verlag (2008)]


This reference list is based on information provided by the publisher or from digital mathematics libraries. Its items are heuristically matched to zbMATH identifiers and may contain data conversion errors. It attempts to reflect the references listed in the original paper as accurately as possible without claiming the completeness or perfect precision of the matching.