Whiley: a Platform for Research in Software Verification

David J. Pearce and Lindsay Groves

Victoria University of Wellington, Wellington, New Zealand
{djp,lindsay}@ecs.vuw.ac.nz

1 Introduction

Prof. Sir Tony Hoare (ACM Turing Award Winner, FRS) proposed the creation of a *verifying compiler* as a grand challenge for computer science [1]. A verifying compiler "*uses automated mathematical and logical reasoning to check the correctness of the programs that it compiles.*" There have been numerous attempts to construct a verifying compiler system, although none has yet made it into the mainstream. Early examples include that of King [2], Deutsch [3], the Gypsy Verification Environment [4] and the Stanford Pascal Verifier [5]. More recently, the Extended Static Checker for Modula-3 [6] which became the Extended Static Checker for Java (ESC/Java) — a widely acclaimed and influential work [7]. Building on this success was JML and its associated tooling which provided a standard notation for specifying functions in Java [8]. Finally, Microsoft developed the Spec# system which is built on top of C# [9].

Both ESC/Java and Spec# build on existing object-oriented languages (i.e. Java and C#) but, as a result, suffer numerous limitations. The problem is that such languages were not designed for use with verifying compilers. Ireland, in his survey on the history of verifying compilers, noted the following [10]:

> *"The choice of programming language(s) targeted by the verifying compiler will have a significant effect on the chances of success."*

Likewise, a report on future directions in verifying compilers, put together by several researchers in this area, makes a similar comment [11]:

> *"Programming language design can reduce the cost of specification and verification by keeping the language simple, by automating more of the work, and by eliminating common errors."*

This talk will introduce Whiley, a programming language designed from scratch in conjunction with a verifying compiler. The intention of this is to provide an open framework for research in automated software verification. The initial goal is to automatically eliminate common errors, such as *null dereferences*, *array-out-of-bounds*, *divide-by-zero* and more. In the future, the intention is to consider more complex issues, such as termination, proof-carrying code and user-supplied proofs. Finally, several works have already been published which focus different aspects of the Whiley system [12, 13, 14, 15, 16].

The Tool. The main tool underlying Whiley is the verifying compiler. This is been in development for over three years, and has become a large (and relatively mature) code base. Numerous student projects have been conducted already based on this compiler, and the hope is to use it for teaching next year. The compiler is released under an open source license (BSD), can be downloaded from `http://whiley.org` and forked at `http://github.com/DavePearce/Whiley/`. Some interesting statistics are available from `http://www.ohloh.net/p/whiley` and a fun demonstration on writing loop invariants is available here: `http://www.youtube.com/watch?v=WwnxHugabrw`. Finally, a prototype Eclipse plugin is available and can be installed via the update site: `http://whiley.org/eclipse`.

References

[1] Tony Hoare. The verifying compiler: A grand challenge for computing research. *Journal of the ACM*, 50(1):63–69, 2003.

[2] S. King. *A Program Verifier*. PhD thesis, Carnegie-Mellon University, 1969.

[3] L. Peter Deutsch. *An interactive program verifier*. Ph.d., 1973.

[4] D. I. Good. Mechanical proofs about computer programs. In *Mathematical logic and programming languages*, pages 55–75, 1985.

[5] D. C. Luckham, S. M. German, F. W. von Henke, R. A. Karp, P. W. Milne, D. C. Oppen, W. Polak, and W. L. Scherlis. Stanford pascal verifier user manual. Technical Report CS-TR-79-731, Stanford University, Department of Computer Science, 1979.

[6] David L. Detlefs, K. Rustan M. Leino, Greg Nelson, and James B. Saxe. Extended static checking. SRC Research Report 159, Compaq Systems Research Center, 1998.

[7] Cormac Flanagan, K. Rustan M. Leino, Mark Lillibridge, Greg Nelson, James B. Saxe, and Raymie Stata. Extended static checking for Java. In *Proc. PLDI*, pages 234–245, 2002.

[8] G. T. Leavens, Y. Cheon, C. Clifton, C. Ruby, and D. R. Cok. How the design of JML accommodates both runtime assertion checking and formal verification. *Science of Computer Programming*, 55(1-3):185–208, March 2005.

[9] Mike Barnett, K. Rustan, M. Leino, and Wolfram Schulte. The spec# programming system: An overview. Technical report, Microsoft Research, 2004.

[10] A. Ireland. A Practical Perspective on the Verifying Compiler Proposal. In *Proceedings of the Grand Challenges in Computing Research Conference*, 2004.

[11] G. T. Leavens, J. Abrial, D. Batory, M. Butler, A. Coglio, K. Fisler, E. Hehner, C. Jones, D. Miller, S. Peyton-Jones, M. Sitaraman, D. R. Smith, and A. Stump. Roadmap for enhanced languages and methods to aid verification. In *Proc. of GPCE*, pages 221–235, 2006.

[12] D. Pearce and J. Noble. Implementing a language with flow-sensitive and structural typing on the JVM. *Electronic Notes in Computer Science*, 279(1):47–59, 2011.

[13] D. J. Pearce. Sound and complete flow typing with unions, intersections and negations. In *Proc. VMCAI*, pages 335–354, 2013.

[14] D. J. Pearce. A calculus for constraint-based flow typing. In *Proc. FTFJP*, page Article 7, 2013.

[15] D. J. Pearce and L. Groves. Whiley: a platform for research in software verification. In *Proc. SLE*, page (to appear), 2013.

[16] D. J. Pearce and L. Groves. Reflections on verifying software with whiley. In *Proc. FTSCS*, page (to appear), 2013.