

Towards Verification of Software Product Lines: The JBook Case Study

Don Batory

Department of Computer Sciences
University of Texas at Austin
Austin, Texas, 78712 U.S.A.
batory@cs.utexas.edu

Egon Börger

Dipartimento di Informatica
Università di Pisa
I-56127 Pisa, Italy
boerger@di.unipi.it

Extended Abstract

Product-lines are used in many industries to reduce product development costs, improve product quality, and increase product variability. Sadly, what distinguishes *software product lines (SPLs)* is the absence of meaningful warranties on its products (programs). There are few results on verifying SPLs [5][7].

Scaling verification to large programs is a long-standing problem. There is a growing community of researchers that believe verification must be intimately integrated with software design and modularity for scaling to occur; verification of programs should not be an after-thought. In this presentation, we explore an approach that suggests how feature modularization can scale verification to product-lines of programs. We bring together results from previously unrelated communities: *Abstract State Machines (ASM)* and *Feature-Oriented Programming (FOP)*. ASM is a rigorous method for program specification and verification. FOP is a design methodology and compositional technology for program synthesis [1][2]. ASM and FOP both use step-wise refinement to construct programs and specifications. Our case study is the 2001 JBook [8] that among other results presented a Java/JVM compilation correctness proof for defining, interpreting, compiling, and executing bytecode for the Java 1.0 language. Among the discoveries of JBook were problems with bytecode verification, under-specification of static initializers (leading to portability problems of Java programs), concurrent initializations could deadlock, and existent Java compilers violated initialization semantics through standard optimization techniques [3]. More recent work examined C# with similar results [4][6].

Although ASM and FOP were conceived independently (their roots trace back to the early 1990s), both use *features* — increments in functionality — as a modularization centerpiece. JBook and FOP use features to modularize grammars and programs in an identical way. In this presentation, we justify this claim and more importantly show how features also modularize JBook theorems (both their statements and proofs). By composing features, complete grammars, programs, and theorems are synthesized.

Our results are not limited to JBook or compilers; they are meaningful in the context of SPLs where each program of an SPL may have its own unique set of verification properties and proofs. Instead of manually verifying individual programs, which is a laborious task, theorem statements and proofs can be synthesized, exactly like other program representations. Generated theorems may then be certified manually or automatically using a proof checker.

References

- [1] D. Batory and S. O'Malley. "The Design and Implementation of Hierarchical Software Systems with Reusable Components". *ACM TOSEM*, October 1992.
- [2] D. Batory, J.N. Sarvela and A. Rauschmayer. "Scaling Step-Wise Refinement". *IEEE TSE*, June 2004.
- [3] E. Börger and W. Schulte. "Initialization Problems for Java". *Software—Concepts & Tools*, 20(4), 1999.
- [4] E. Börger, G. Fruja, V. Gervasi and R. Stärk. "A High-Level Modular Definition of the Semantics of C#". *Theoretical Computer Science*, Vol. 336 #2-3, 2005.
- [5] K. Czarnecki and K. Pietroszek. "Verification of Feature-Based Model Templates Against Well-Formedness OCL Constraints". *GPCE 2006*.
- [6] N.G. Fruja. "Type Safety of C# and .NET CLR", ETH Zurich, 2006.
- [7] S. Krishnamurthi and K. Fisler. "Modular Verification of Collaboration-Based Software Designs". *FSE 2001*.
- [8] R.F. Stärk, J. Schmid and E. Börger. *Java and the Java Virtual Machine: Definition, Verification, Validation*. Springer-Verlag, 2001.