Formal Software Development using Generic Development Steps

Axel Dold
Fakultät für Informatik
Universität Ulm
D-89069 Ulm, Germany
Fax: ++49 731-50-24119
dold@ki.informatik.uni-ulm.de

Abstract

This talk is concerned with a mechanized formal treatment of the transformational software development process in a unified framework. As a formal vehicle, the specification and verification system PVS [7] is utilized to integrate development steps and development methods from different existing transformational approaches (for example, PROSPECTRA [6], KIDS [9], CIP [1,5,8], Bird-Meertens [2]). Integration comprises the formalization (that is, the implementation in the PVS specification language), the verification, and the correct application of the generic development steps to specific problems. Transformations of different kind and complexity have been integrated into this framework comprising the whole development process: problem-solving strategies encoding general algorithmic paradigms such as *global-search* and *divide-and-conquer* [10], transformations for the modification of functional specifications such as transformations from the Bird-Meertens formalism like *fusion* or *Horner's rule*, transformation steps for optimizing recursive functions [3,8], transformations on the level of procedural programs, and implementations of data structures.

All software artifacts are represented generically within parameterized PVS theories which specify the semantic requirements on parameters by means of assumptions and define the result of the transformation. The parameters reflect the design decisions realized by the transformation. Based on the semantic requirements, correctness of the generic step can be proved. Application of such a development step to a given problem is then carried out by providing a concrete instantiation for the parameters and verifying that it satisfies the theory requirements.

The problem-solving strategies have been organized by means of taxonomies (that is, hierarchies of specializations for specific applications and data structures). This approach, inspired by concepts from AI knowledge representation formalisms, greatly improves the applicability of the transformations and leads to an elegant structure of the algorithmic knowledge captured in the development steps.

As a basis for the realization of refinement hierarchies and for establishing a formal software development methodology in PVS, a notion of refinement is presented which is based on correct partial instantiations of parameterized theories.

We have illustrated the usability of the approach by many examples of different complexity including the derivation of an interpreter for Prolog, the implementation of a compiler symbol table, and as a larger case-study: the construction of a compiler program from a given compiling specification specifying the translation of a Common-Lisp-like language into a stack intermediate language (see [4]). An excerpt from a formal derivation example will be presented in this talk.

References

[1] Klaus Achatz and Helmuth Partsch. From Descriptive Specifications to Operational ones: A Powerful Transformation Rule, its Applications and Variants. Technical Report 96-13, Universität Ulm, December 1996.

[2] Richard S. Bird. The Promotion and Accumulation Strategies in Transformational Programming. *ACM - Transactions on Programming Languages and Systems*, 6(4):487–504, 1984.

[3] Eerke A. Boiten. Improving Recursive Functions by Inverting the Order of Evaluation. *Science of Computer Programming*, 18:139–179, 1992.

[4] Axel Dold. *Formal Software Development using Generic Development Step*. PhD thesis, Universität Ulm, Fakultät für Informatik, 1999. to appear.

[5] The CIP System Group. *The Munich Project CIP - Volume II: The Program Transformation System CIP-S*. LNCS 292. Springer-Verlag, 1987.

[6] Berthold Hoffmann and Bernd Krieg-Brückner (Eds.). *Program Development by Specification and Transformation - The PROSPECTRA Methodology, Language Family, and System*, volume 680 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[7] S. Owre, J. Rushby, N. Shankar, and F. von Henke. Formal Verification for Fault-Tolerant Architectures: Prolegomena to the Design of PVS. *IEEE Transactions on Software Engineering*, 21(2):107–125, February 1995.

[8] Helmuth A. Partsch. *Specification and Transformation of Programs - A Formal Approach to Software Development*. Texts and Monographs in Computer Science. Springer-Verlag, 1990.

[9] Douglas R. Smith. KIDS - A Semi-Automatic Program Development System. *IEEE - Transactions on Software Engineering, Special Issue on Formal Methods in Software Engineering*, 16(9):1024–1043, September 1990.

[10] Douglas R. Smith and Michael R. Lowry. Algorithm Theories and Design Tactics. *Science of Computer Programming*, 14(2-3):305–321, 1990.