Programming Language Elements for Correctness Proofs

Gergely Dévai

The study of formal methods to reason about program properties is getting a more and more important research area, as a considerable part of a software product's lifecycle is testing and bugfixing. The theoretical basis — such as formal programming models and reasoning rules [1, 2, 3, 5, 6, 7] — has been developed so far, but these are rarely used in industry [10]. The main reason for this fact is that formally proving a program property usually takes much more time than writing the program itself.

The goal of this research is to use programming language elements to make the construction of these proofs easier. The basic idea is to develop a new programming language where the source code contains the formal specification and the correctness proof of the implementation. The proofs are built up using *stepwise refinement* [4, 8, 9], as this technique provides *correctness by construction* [12], and also helps the programmers to make the right decisions during software development. The compiler of the language has to check the soundness of the proof steps and to generate the program code in a target language using the information of the proof.

Similarly to programs, proofs also contain schematic fragments. These can be managed efficiently using *"proof templates"* that have the same role in proof construction as procedures have in traditional program development. This leads to a special kind of *generative programming* [13]: by the instantiation of the templates a proof is constructed (and checked) in compile time and from the proof a target language program is generated which automatically fulfils all the requirements stated in the specification.

In this paper we show how language elements can be used to make specification statements easy to understand and to write, we introduce the basic refinement rules of the model and show how templates can be used to construct the proof. The algorithms used by the compiler to check the soundness of the proof and to generate the target language program are discussed too.

References

[1] C.A.R. Hoare. An axiomatic basis for computer programming, *Commun. ACM*, 12, 10, 1969, 0001-0782, 576–580, `http://doi.acm.org/10.1145/363235.363259`, ACM Press, New York, NY, USA.

[2] E.W. Dijkstra. A Discipline of Programming, *Prentice-Hall*, 1976, DIJ e 76:1 1.Ex,

[3] F. Kröger. Temporal Logic of Programs, *Springer*, 1987, Berlin, Heidelberg.

[4] J.M. Morris. A theoretical basis for stepwise refinement and the programming calculus, *Sci. Comput. Program.*, 9, 3, 1987, 0167-6423, 287–306, `http://dx.doi.org/10.1016/0167-6423(87)90011-6`, Elsevier North-Holland, Inc., Amsterdam, The Netherlands.

[5] K.M. Chandy and J. Misra. Parallel Program Design, A Foundation, *Addison-Wesley*, 1988.

[6] Z. Horváth. Párhuzamos programok relációs programozási modellje, ELTE, TTK, Informatika Doktori Iskola, 1996.

[7] Z. Horváth, T. Kozsik, and M. Tejfel. Proving Invariants of Functional Programs, *Proceedings of the Eighth Symposium on Programming Languages and Software Tools*, 115-127, 2003.

[8] C. Morgan. Programming from specifications, Second, *Prentice Hall International (UK) Ltd.*, 1994.

[9] J.-R. Abrial. The B-book: assigning programs to meanings, *Cambridge University Press*, New York, NY, USA, 1996, 0-521-49619-5.

[10] J.P. Bowen and M.G. Hinchey. Ten commandments revisited: a ten-year perspective on the industrial application of formal methods, *FMICS '05: Proceedings of the 10th international workshop on Formal methods for industrial critical systems*, 2005, 1-59593-148-1, 8–16, lLisbon, Portugal, http://doi.acm.org/10.1145/1081180.1081183, ACM Press, New York, NY, USA.

[11] D. Pavlovic and D. Smith. Software development by refinement, In *UNU/IIST 10th Anniversary Colloqium*, Formal Methods at the Crossroads: From Panaea to Foundational Support. Springer-Verlag, 2003., 2003,
`citeseer.ist.psu.edu/pavlovic03software.html`.

[12] J. McDonald and J. Anton. SPECWARE - Producing Software Correct by Construction, *Kestrel Institute Technical Report*, KES.U.01.3., March 2001., 2001.

[13] M. Czarnecki and U. Eisenecker. Generative Programming: Methods, Tools, and Applications, *Addison-Wesley*, CZA k 00:1 1.Ex, 2000.