Prototype Environment for Refactoring Clean Programs

Rozália Szabó-Nacsa, Péter Diviánszky and Zoltán Horváth

We present here the prototype of an interactive environment where one can incrementally carry out programmer-guided meaning-preserving program transformations in functional languages. We discuss an alternative approach to the problems of storing and extracting the syntactic and also the static semantic information in order to be enough to perform the desired transformations. In our approach the program to be redesigned is stored in a relational database.

Several transformation case studies will help us to demonstrate how this database can be used to transform programs, check the preconditions and make compensation steps to ensure correct transformations.

We also show an interactive environment which will help the programmer to choose the appropriate refactoring step and its parameters. During redesign process the programmer is faced with one of the selected "views" extracted from the database. Different transformations can be carried out on different views, depending on which view is preferable for the programmer and / or which view is more suitable for the given transformation.

References

[1] Li, H., Reinke, C., Thompson, S.: Tool Support for Refactoring Functional Programs, Haskell Workshop: Proceedings of the ACM SIGPLAN workshop on Haskell, Uppsala, Sweden, Pages: 27-38, 2003.

[2] Fóthi, Á., Horváth, Z., Nyéky-Gaizler, J.: A Relational Model of Transformation in Programming, Proceedings of the 3rd International Conference on Applied Informatics, Eger-Noszvaj, Hungary, Aug. 26-28, 1997. 335-349.

[3] Plasmeijer, R., Eekelen, M.: Concurren Clean Language Report, Technical Report CSI-R9816, Computing Science Institute, University of Nijmegen, 1998.

[4] Fowler, M., Beck, K., Brant, J., Opdyke, W., Roberts, D.: Refactoring: Improving the Design of Existing Code, Addison-Wesley, 1999.

[5] Martin Fowler's refactoring site, `www.refactoring.com`

[6] de Mol, M., van Eekelen, m., Plasmeijer, R.: SPARKLE: A Functional Theorem Prover, International Workshop on the Implementation of Functional Languages, IFL 2001, Selected Papers, Springer-Verlag, LNCS 2312, pages 55-71.

[7] Horváth, Z., Kozsik, T., Tejfel, M.: Verifying invariants of abstract functional objects - a case study 6th International Conference on Applied Informatics, Eger, Hungary January 27-31, 2004.