Language for Erlang Refactoring

Dániel Horpácsi

Department of Programming Languages and Compilers, Eötvös Loránd University

daniel-h@elte.hu

A refactoring is a program transformation that improves non-functional attributes of source code without modifying its external behaviour. Behaviour-preserving transformations on large-scale software systems are performed with the assistance of refactoring tools in order to avoid incomplete or inconsistent modifications. Refactoring tools implement parsing, static analysis, pretty-printing, as well as different kinds of transformations for a specific language.

In most cases, refactorings are specified informally on the level of the programming language concepts, then they are implemented on a low level of abstraction, incorporating the intermediate inner representation used in the refactoring tool, such as the abstract syntax tree or the semantic program graph. This is problematic in many ways: 1) there is a considerable gap between the abstraction level of the specification and the implementation which raises issues regarding correctness, 2) due to the low abstraction level of implementation, refactoring definitions contain large amounts of boilerplate, representation-specific code that manipulates the model and 3) only the developers of the refactoring tool have the required knowledge to define transformations.

Refactoring transformations should be defined with language-level concepts, so that they do not contain representation-specific details, making the users of the language able to craft their own refactorings. The definitions are meant to melt the specification and the implementation into one description, which is on the other hand verifiable as well as executable. There is a rich related work in the field of high-level program transformation [**?**] and refactoring definition [**?**, **?**], but neither of the existing solutions support definitions that are executable, verifiable, and applicable at the same time. In this paper, we introduce a novel formalism, implemented as a domain specific language (DSL), for defining refactorings for the programming language Erlang [**?**, **?**]. To demonstrate the applicability of our approach, we show how to define a number of well-known refactorings in the transformation language.

References

[1] I. Bozó, V. Fördős, D. Horpácsi, Z. Horváth, T. Kozsik, J. Kőszegi, and M. Tóth. *TFP '14*, Refactorings to Enable Parallelization, pages 104–121. Springer International Publishing, Cham, 2015.

[2] M. Bravenboer, A. van Dam, K. Olmos, and E. Visser. Program Transformation with Scoped Dynamic Rewrite Rules. *Fundam. Inf.*, 69(1-2):123–178, July 2005.

[3] F. Cesarini and S. Thompson. *Erlang Programming*. O'Reilly Media, Inc., 2009.

[4] H. Li and S. Thompson. A Domain-specific Language for Scripting Refactorings in Erlang. In *Proceedings of FASE'12*, pages 501–515, Berlin, Heidelberg, 2012. Springer-Verlag.

[5] M. Schaefer and O. de Moor. Specifying and Implementing Refactorings. *SIGPLAN Not.*, 45(10):286–301, Oct. 2010.