Static analysis and visualisation of Erlang generic server behaviours

Mátyás Béla Kuti, István Bozó, Melinda Tóth

Eötvös Loránd University, Faculty of Informatics, Department of Programming Languages and Compilers

{`kumtaai,bozoistvan,tothmelinda`}`@caesar.elte.hu`

Understanding and maintaining the source code of an industrial-scale software product is hard and time-consuming, and it is getting more difficult when the software implements parallel, concurrent or distributed computations and behaviours. Static source code analysis tools support program comprehension through detecting dependencies and relations among small (like functions and processes) or large (modules and components) software elements.

Tools that support the software development life-cycle (such as code comprehension tools, refactoring tools, visualisation tools, test selection tools, test coverage checkers, etc.) are frequently used by the industry. These tools can reduce the time of bug fixing, code change requests, new feature requests and they also help to decrease the number of faults in the system.

Static analyser tools extract information from the source code without executing the program. Static analysis is not straightforward for sequential programs and it is getting more difficult when we are to analyse complex concurrent or distributed software.

Erlang [**?**] is a concurrent functional programming language that was designed for developing telecommunication systems in the 1980s. RefactorErl [**?**] is a tool for Erlang, that aims to support meaning preserving source code transformations and code comprehension. It represents the source code in a so-called Semantic Program Graph (SPG). The SPG contains the abstract syntax tree of the source code, and in addition semantic nodes and edges representing semantic information. For instance, functions and function calls and the connections generated by the usage of the built in communication primitives of the language, etc.

Erlang provides patterns (so called behaviours) for concurrent and distributed application design. There are several behaviours in the Erlang/OTP [**?**] library, such as generic servers, finite state machines, supervisors, etc.

The main focus of our paper is the extension of the Semantic Program Graph with application specific semantic information. We extend the communication and process semantic layer found in the SPG with the results of generic server analysis. We also define a separate graph, a view of a static interaction model (communication and message handling) of the generic server behaviour.

References

[1] I. Bozó, D. Horpácsi, Z. Horváth, R. Kitlei, J. Kőszegi, M. Tejfel, and M. Tóth. Refactorerl – source code analysis and refactoring in Erlang. In *In proceeding of the 12th Symposium on Programming Languages and Software Tools, Tallin, Estonia*, 2011.

[2] Logan, M., Merritt, E., and Carlsson, R.: *Erlang and OTP in Action*, Manning Publications Co., 2010. ISBN 9781933988788.