Executable Semantics for Erlang

Judit Kőszegi

Department of Programming Languages and Compilers, Eötvös Loránd University

koszegijudit@elte.hu

Formal definition of programming language semantics is crucial to formally reason about program properties. Traditional program verifiers are based on either Hoare logic, separation logic or dynamic logic. Semantics definitions in these logics are rather complex and their implementations provide no practical way for the validation of the semantics itself. The recently introduced matching logic [**?**] merges the advantages of the verification-focused semantics definition methods while it is more intuitive as it closely relates to how the program is executed (similarly to operational semantics). Its implementation, the $\mathbb{K}$ [**?**] semantics framework supports formal definition of languages, as well as execution, model checking and verification of programs.

For many years we have worked on research projects related to the Erlang programming language, and from the very beginning there was a demand to not only test, but formally verify some components of our tools. To this end, we need a formal semantics definition for Erlang along with a verification logic. In the related work, we can find an almost complete small-step operational semantics defined by Fredlund [**?**], but it is coupled together with modal μ-calculus which is unsuitable for formal verification in practice due its complexity. Therefore, we decided to define the semantics of the language with matching logic in the $\mathbb{K}$ framework, which proves itself to be an all-in-one solution: after specifying the formal semantics of the language, we get the verification methods and toolset for free. The starting point for our definition is the definition composed by Fredlund, but the operational semantic rules have to be rephrased in matching logic, and we also extend it by adding language features missing from the original definition. This paper introduces the challenges in defining matching logic semantics of Erlang, and shows practical applications: we use the semantics for symbolic program verification and checking the correctness of refactoring transformations [**?**].

References

[1] L.-A. Fredlund. *A Framework for Reasoning about Erlang code.* PhD thesis, Royal Institute of Technology, Stockholm, Sweden, 2001.

[2] K framework. `http://www.kframework.org`. Accessed April, 2016.

[3] A. Arusoaie, D. Lucanu, and V. Rusu. A Generic Framework for Symbolic Execution: Theory and Applications. Research Report RR-8189, Inria, Sept. 2015.

[4] D. Horpácsi, J. Kőszegi, and S. Thompson. Towards Trustworthy Refactoring in Erlang. Preprint Workshop Paper VPT'16