Targeted static fault localization in Erlang programs

Zsófia Erdei, Melinda Tóth, István Bozó

ELTE, Eötvös Loránd University

zsanart@inf.elte.hu, toth_m@inf.elte.hu, bozo_i@inf.elte.hu

Static source code analysis techniques may help the programmers in various tasks: code comprehension, testing, debugging, etc. They often need to reproduce executions that result in faulty behaviour. Program analysis techniques with symbolic execution can help to solve this task. In this paper we propose a method to select an appropriate execution path from the static control-flow graph that may lead to a given runtime error in Erlang software. We build our tool on the RefactorErl static analyser framework.

Fault localization is the act of identifying the locations of faults in a program. Even when bugs in software are discovered due to some faulty behavior (e.g. a runtime error occurs), finding the location of the fault is a non-trivial task. Error detection mechanisms are vital for building highly reliable systems. Fault localization is one of the most time consuming, and expensive part of software development and maintenance. Given the size and complexity of large-scale software systems today, manual fault localization becomes more and more futile, so effective automatic methods are needed.

In a concrete execution, a program is evaluated on a specific input and a single control-flow path is explored. Symbolic execution [1, 2] uses unknown symbolic variables in evaluation, allowing to simultaneously explore multiple paths that a program could take under different inputs. We can use symbolic execution to help us in fault localization. We target to find an execution path in the program, the ”error path”, that may result in a runtime error in a given point of the program. Thus we build a direct symbolic execution engine for a given execution path in the Erlang programs based on the RefactorErl framework [3]. We are using the SMT solver of Z3 [4] to solve the constraints that we gather during our analysis.

We can use symbolic execution to help us in fault localization. We target to find an execution path in the program, the ”error path”, that may result in a runtime error in a given point of the program. Thus we build a direct symbolic execution engine for a given execution path in the Erlang programs based on the RefactorErl framework [3]. We are using the SMT solver of Z3 [4] to solve the constraints that we gather during our analysis.

References

[1] James C. King. Symbolic execution and program testing. *Commun. ACM*, 19(7):385–394, July 1976.

[2] Roberto Baldoni, Emilio Coppa, Daniele Cono D'elia, Camil Demetrescu, and Irene Finocchi. A survey of symbolic execution techniques. *ACM Comput. Surv.*, 51(3), May 2018.

[3] M. Tóth and I. Bozó. Static analysis of complex software systems implemented in erlang. Central European Functional Programming Summer School – Fourth Summer School, CEFP 2011, Revisited Selected Lectures, Lecture Notes in Computer Science (LNCS), Vol. 7241, pp. 451-514, Springer-Verlag, ISSN: 0302-9743, 2012.

[4] Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.