8th International Conference on Applied Informatics
Eger, Hungary, January 27–30, 2010.

Slicing Erlang programs*

István Bozó, Melinda Tóth, Zoltán Horváth

Department of Programming Languages and Compilers,
Eötvös Loránd University, Budapest, Hungary
e-mail: {bozo_i,toth_m,hz}@inf.elte.hu

Abstract

Program slicing is the most well-known approach in impact analysis when the programmer tries to detect those program components which are affected by a change. Measuring the impact of a change by detecting those program parts which are affected by a source code transformation could help to reduce the number of test cases must be performed during a regression test. Considering that regression testing is the most expensive part of a software development cycle and during the lifetime of a software product certain changes could be often performed on its source code, reducing the number of test cases with impact analysis could be an elemental part of software development.

This paper focuses on programs written in Erlang. Erlang is a dynamic functional programming language, where most of the program bugs can not be detected in compile time, just in run-time, thus testing after a source code transformation is necessary. Testing Erlang programs with property based testing using QuickCheck properties is popular in the industry. Thus we focus on selecting QuickCheck properties using Erlang program slicing. The paper gives the definition of the Control Flow Graph (CFG) of Erlang programs, the definition of Erlang program slices and the program slicing mechanism using the CFG and the Erlang Behaviour Dependency Graph. When the resulting program slice does not contain a property, the impact of the change does not affect it, thus it is not necessary to retest the property during the regression testing.

References

[1] M. Weiser., Program slices: Formal, psychological, and practical investigations of an automatic program abstraction method. *PhD thesis, University of Michigan, Ann Arbor, MI, 1979.*

[2] M. Harman, D. W. Binkley, and S. Danicic., Amorphous program slicing. *Journal of Systems and Software, 68(1):45-64, Oct. 2003.*

*Supported by TECH_08_A2-SZOMIN08, ELTE IKKK, and Ericsson Hungary

[3] M. Harman and S. Danicic., Amorphous program slicing. *In5th IEEE International Workshop on Program Comprenhesion (IWPC-97), pages 70-79, Dearborn, Michigan, USA, May 1997. IEEE*

[4] D. Binkley, S. Danicic, T. Gyimóthy, M. Harman, Á. Kiss, and L. Ouarbya., Formalizing executable dynamic and forward slicing. *In Proceedings of the 4th IEEE International Workshop on Source Code Analysis and Manipulation (SCAM 2004), pages 43-52, Chicago, Illinois, USA, September 15-16, 2004. IEEE Computer Society*

[5] B. Korel and J. Laski., Dynamic program slicing. *Information Processing Letters, 29(3):155-163, October 1988.*

[6] J. Ferrante, K. J. Ottenstein, and J. D. Warren., The program dependence graph and its use in optimization. *ACM Transactions on Programming Languages and Systems, 9(3):319-349, July 1987.*

[7] S. Horwitz, T. Reps, and D. Binkley., Interprocedural slicing using dependence graphs. *ACM Transactions on Programming Languages and Systems, 12(1):3546, January 1990.*

[8] H. Agrawal and J. Horgan., Dynamic program slicing. *In Proceedings of the ACM SIGPLAN '90 Conference, 1990.*

[9] K. Ottenstein and L. Ottenstein., The program dependence graph in software development environments. *In Proceedings of the ACM SIGSOFT/SIGPLAN Software Engineering Symposium on Practical Software Development Environments, pages 177.*