Dynamic Slicing of Programs Compiled for the Java Virtual Machine

Attila Szegedi and Tibor Gyimóthy

In this paper, we present a technique for obtaining dynamic slices of programs compiled for the Java virtual machine. The presented technique is independent of source language, therefore works for programs written in any language that can be compiled to Java virtual machine bytecode. In contrast with existing published techniques [1] that require a customized Java compiler (which also implies access to the source code and being limited to the Java language) our approach works with programs compiled with arbitrary third party compilers designed for arbitrary source level language. As a consequence, our method does not require access to the source code during any point of the slicing process. However, we still retain the ability to express the slicing criterion and the resulting slice in terms of source code locations using the line number information present in compiled code.

We do not instrument the source nor the compiled bytecode, but instead use a special instrumented virtual machine. An advantage of the approach is that we can successfully track dependencies generated through execution of third-party library code, standard Java library code, and even code that was dynamically generated during program execution (dynamically generated code is an ever more frequently used Java technique), as well as operations performed in virtual machine's native code (i.e. object cloning). Since our ultimate goal is covering all of the internal dependencies that can possibly occur during the execution of a program in a Java virtual machine, we cover specific aspects like reverse interprocedural flow dependencies (from callees to callers) introduced by catching exceptions in caller methods thrown by their callees, inter-thread notifications, and even limited ability for tracking dependencies in external native code called through the JNI interface. The presented execution history format provides full-fidelity representation for multithreaded execution, which is also a natural feature of the Java virtual machine that must be fully supported. We also present the static preprocessing steps for slicing that are specific to the method: constructing the left-hand-side expressions in assignment instructions without relying on source code, as well as the control flow calculations that take into account exception handlers. The slicing algorithm used is a variant of the forward global method for computing backward dynamic slices based on work presented in [2].

References

[1] F. Umemori, K. Konda, R. Yokomori, K. Inoue: Design and Implementation of Bytecode-based Java Slicing System, Proceedings of the Third IEEE International Workshop on Source Code Analysis and Manipulation, pages 108- 117. Amsterdam, The Netherlands, September 26-27, 2003.

[2] Beszédes, Á., Gergely, T., Szabó, Zs. M., Csirik, J., and Gyimóthy T.: Dynamic Slicing Method for Maintenance of Large C Programs, Proceedings of the 5th European Conference on Software Maintenance and Reengineering (CSMR 2001), pages 105-113. Lisbon, Portugal, March 14-16, 2001.