Extended Code Coverage for AspectJ-based Runtime Verification Tools

Many runtime verification tools for the Java virtual machine rely on aspect-oriented programming, particularly on AspectJ, to weave the verification logic into the observed program. However, AspectJ imposes several limitations on the verification tools, such as a restricted join point model and the inability of weaving certain classes, particularly the Java and Android class libraries. In this paper, we show that our domain-specific aspect language DiSL can overcome these limitations. While offering a programming model akin to AspectJ, DiSL features an extensible join point model and ensures weaving with complete bytecode coverage for Java and Android. We present a new compiler that translates runtime-verification aspects written in AspectJ to DiSL. Hence, it is possible to use existing, unmodified runtime verification tools on top of the DiSL framework to bypass the limitations of AspectJ. As a case study, we show that the AspectJ-based runtime verification tool JavaMOP significantly benefits from the automated translation of AspectJ to DiSL code, gaining increased code coverage. Thanks to DiSL, JavaMOP analyses are able to unveil violations in the Java class library that cannot be detected when using AspectJ.

Author: Walter Binder (walter.binder@usi.ch)