Infinitely Often Testing (Extended Abstract)

Willem Visser

Department of Computer Science, Stellenbosch University, South Africa
visserw@sun.ac.za

From the perspective of industry, formal methods over-promise and under-deliver. Theoretical computer scientists love the notion of proving programs correct, but have slowly come round to the realization that promises in grant proposals aren't the same as delivering in the real world. Essentially we started seeing a slow erosion of the importance of the notion of soundness; completeness was dropped long before. The ideal of showing that programs behave according to their specification, became the reality of finding situations where they don't. This maps perfectly onto an expensive activity well known to industry, namely software testing. This presentation looks at the happy marriage of techniques from formal methods and software testing. Software testing is expensive since it is time-consuming to derive tests to adequately cover the software's behavior. Techniques from formal methods allow one to generate tests automatically (hence reducing costs) and systematically (hence increasing the likelihood of discovering errors). We look at the use of software model checking to find errors in complex code, and specifically, consider the evolution of one of the world's most popular model checkers, JavaPathFinder (JPF). One of the core techniques in JPF is symbolic execution that, although introduced in the early seventies, has recently made a big comeback in the testing world. We discuss the reasons why it took this long for such a powerful technique to become popular (again) and how it is used within JPF. In addition we discuss some of the new advances in symbolic execution and how it is used for bug finding and test generation. Finally, we consider some of the new challenges facing the automated testing field and how formal techniques can be applied to address them.

A. Cerone and P. Pihlajasaari (Eds.): ICTAC 2011, LNCS 6916, p. 6, 2011.