Mining Models

Andreas Zeller

Saarland University – Computer Science, Saarbrücken, Germany
zeller@cs.uni-saarland.de

Abstract. Modern Model Checking techniques can easily verify advanced properties in complex software systems. Specifying these models and properties is as hard as ever, though. I present techniques to extract models from legacy systems—models that are precise and complete enough to serve as specifications, and which open the door to modular verification.

Automated validation of software systems has made tremendous progress over the past decade. But all validation, be it static, dynamic, or manual, depends on a *specification* to be validated against. Where shall we get these specifications from? It is easy to specify that a pointer be not null, that a buffer shall not overflow, or that a number may stay within a specific range (and it is hard enough to validate such claims!). But if we want to validate more complex patterns of behavior, we will have to deal with *specifying* these patterns first. This is not so much a technical challenge, but a *social* challenge: We can easily incorporate our validation knowledge into automatic verifiers, which can then be used as black boxes even by laymen. But how shall we teach programmers how to specify behavior—at a time when entire domains like the Web have programming languages designed by amateurs and programs written by amateurs? Our only luck so far is that the exploits are written by amateurs as well.

One attempt to improve the situation is to *mine models* from existing systems—models that are precise and concise enough that they can serve as *specifications* for building, validating, or even synthesizing new systems. This is motivated by two key observation: First, it is easier to read (and possibly extend) a given specification rather than develop one from scratch. Second, the past 40 years of programming have encoded lots of knowledge into existing programs that is in daily usage, and possibly a more trustworthy source than any specification I can write from scratch.

Specification mining is hard, however. First, we need *accurate* approaches: Static approaches suffer from *overapproximation:* they encode more behavior than is actually possible. Dynamic approaches suffer from *underapproximation,* as they can learn only from a finite number of executions. Second, the *language* by which we express specifications needs to be *general,* such that it can be easily understood, yet *specific* for the project at hand, such that we can exploit the abstractions of the domain. Third, there is an unlimited number of properties one can mine; and we need to find out which of these are *relevant* for the functionality—and for the programmer. In this SPIN 2012 invited keynote, I present some solutions for these challenges and highlight the potential of model mining, up to a vision of seamless integration of specification and programming.

Reference

1. Zeller, A.: Specifications for Free. In: Bobaru, M., Havelund, K., Holzmann, G.J., Joshi, R. (eds.) NFM 2011. LNCS, vol. 6617, pp. 2–12. Springer, Heidelberg (2011)

A. Donaldson and D. Parker (Eds.): SPIN 2012, LNCS 7385, p. 23, 2012.