

Counterexample Detection, Core Extraction, and Simulation: Three Analyses Applied to a Flash File System

Eunsuk Kang

In this talk, I will describe three general forms of analysis that are available in our tool, the Alloy Analyzer, and our experience applying them in the POSIX pilot project. **Counterexample detection** provides an invaluable aid for identifying subtle errors that lead to a property violation. **Core extraction** highlights portions of a model that are used to establish the correctness of an assertion (also called “coverage”). A core displaying an unexpected or insufficient coverage points to an overconstraint(s) that causes an assertion to hold vacuously true. **Simulation** offers useful insights into system behaviours that may otherwise be overlooked by the exhaustive nature of model checking or theorem proving.

Alloy is a declarative modelling language for expressing structural properties and behaviours of a system. The Alloy Analyzer is equipped with a powerful model finder that can generate an invariant instance, simulate an operation, or search for a counterexample to an assertion. I will show that within the context of model finding, the three analyses—counterexample detection, core extraction, and simulation—fit together in a simple and convenient way. I will describe how the analyses have aided us in the POSIX pilot project—by allowing us to detect design flaws early on and manage the types of complexity that arise in a flash file system.

I will also talk about various challenges that we faced during the project. These analyses are not without weaknesses: counterexample detection suffers from state space explosion as a model grows more complex; the portion of a model highlighted by a core is sometimes too coarse to be useful to the user; and a large, complex simulation trace is difficult to visualize. I will discuss how we plan to tackle each of these challenges in future work.