Abstract

There is a new generation of analysis-based tools to assist programmers in assuring software at scale. These tools emphasize program properties that tend to defy traditional testing and inspection—typically properties that involve non-determinism or that may have a non-local character, in the sense that there may be no single place in the code associated with errors. Concurrency errors are the perfect storm of non-determinism and non-locality, and so have received attention from analysis and verification researchers for some time. Additionally, concurrency is of increasing importance because software exploitation of multicore processors is an emerging gateway to continued movement up the curve of Moore's Law.

In this talk, we survey some recent progress in advanced software assurance tools, with emphasis on concurrency, including both static and dynamic analysis. Our focus is on the effective exploitation of design intent to assist developers in achieving verification related to state consistency and thread confinement. We consider lock-based approaches, using both lexical-style locking, as with Java's “synchronized,” and dynamic lock acquisition and release, as with Doug Lea’s java.util.concurrent library. We also consider policy-based approaches that avoid explicit locking, as used in most GUI frameworks and in many simulation and data management systems.

We summarize the results of a number of field trials in major commercial companies to assess our ideas related to adoptability by working developers and scalability to realistic large systems.