Toward a Shared Infrastructure for Code Checking, Angelic Execution, Debugging, and Synthesis (Invited Talk)

Emina Torlak
University of California, Berkeley, USA
emina@alum.mit.edu

Abstract

Decision procedures have helped automate key aspects of programming: coming up with a code fragment that implements a desired behavior (synthesis); establishing that an implementation satisfies a desired property (code checking); locating code fragments that cause an undesirable behavior (debugging); and running a partially implemented program to test its existing behaviors (angelic execution). Each of these aspects is supported, at least in part, by a family of formal tools. Most such tools are built on infrastructures that are tailored for a particular purpose, e.g., Boogie for verification and Sketch for synthesis. But so far, no single infrastructure provides a platform for automating the full spectrum of programming activities, making it hard to share advances (in encodings, abstractions, and domain-specific optimizations) across different families of tools.

This talk outlines a path toward a shared infrastructure for computer-aided programming, drawing lessons from a decade of collective experience in using relational constraint solvers to design and analyze software. We start with a relational view of the heap pioneered in the early work on the Alloy language and Analyzer. We then derive an encoding of programs in the bounded relational logic of Kodkod, which extends Alloy with support for partial models. Next, we show by example how to use such an encoding to build a program checker, an angelic oracle, an automated debugger, and a template-based synthesis tool. We conclude by discussing some of the challenges involved in lifting the low-level commonalities between these systems into an efficient infrastructure for prototyping, embedding, and synthesis of programming tools.