Using Types for Software Verification

Ranjit Jhala

University of California at San Diego

Traditional software verification algorithms work by using a combination of Floyd-Hoare Logics, Model Checking and Abstract Interpretation, to infer (and check) suitable program invariants. However, these techniques are problematic in the presence of complex (but ubiquitous) constructs like generic data structures, first-class functions.

We demonstrate that modern type systems are capable of the kind of analysis needed to analyze the above constructs, and we use this observation to develop Liquid Types, a new static verification technique which combines the complementary strengths of Floyd-Hoare logics, Model Checking, and Types.

We start in a high-level functional setting (Ocaml), and show how liquid types can be used to statically verify properties ranging from memory safety to data structure “correctness”. We will then show how, by carefully reasoning about pointer arithmetic and aliasing, we can profitably use Liquid Types to verify low-level imperative (C) programs.

This presentation is based on joint work with Patrick Rondon and Ming Kawaguchi.