TITLE: "Synthesizing and Repairing Expressions using Types and Weights"

ABSTRACT: Developing modern software typically involves composing functionality from existing libraries. This task is difficult because libraries may expose many methods to the developer. To help developers in such scenarios, in the first part of the talk, we present a technique that synthesizes and suggests valid expressions of a given type at a given program point. As the basis of our technique we use type inhabitation for lambda calculus terms in long normal form. We introduce a succinct representation for type judgements that merges types into equivalence classes to reduce the search space, then reconstructs any desired number of solutions on demand. Furthermore, we introduce a method to rank solutions based on weights derived from a corpus of code. We implemented the algorithm and deployed it as a plugin for the Eclipse IDE for Scala. We show that the techniques we incorporated greatly increase the effectiveness of the approach. In the second part of the talk, we briefly introduce a technique that repairs a broken expression and returns a set of correct expressions that follow the structure of the broken expression as close as possible.