Keynote

SCAM 2009

How Analysis Can Hinder Source Code Manipulation
-- And What to Do about It

Michael Ernst
Computer Science & Engineering
University of Washington

Abstract
Source code analysis is a prerequisite to source code manipulation. An understanding what the code does -- that is, a model of its behavior -- is needed in order to propose or to verify a transformation, to enable subsequent analyses, and for other purposes. The research community is justifiably proud of its many successful analysis techniques and applications.

However, in some cases an analysis can hinder source code manipulation. Simple examples are a type system, style checker, or analysis that forbids, or cannot handle, certain code idioms. Every programmer is familiar with adjusting coding style to the limitations of a language, framework, build system, etc. Similar limitations apply to automated program manipulation.

Related problems arise when analyzing legacy code. A programmer may have chosen among multiple designs with similar qualities. Later, when an analysis reveals a reason that one of them was preferable, the cost of reworking the program may be prohibitive, particularly if the transformation is not justified on correctness grounds.

I will explore these and related problems in the context of type inference, type checking, and testing -- drawing connections and highlighting differences among them. I will illustrate both the strengths of these activities and also some ways that they are less useful than they could be. I will suggest lifting some of their strictures, either temporarily or permanently, in an effort to give programmers the best of all worlds.