



Compiling with Types and Flows

Allyn Dimock
Harvard University
Friday March 8
3:00PM
Olsen 311

Abstract

Typed intermediate languages for compilers provide many advantages over untyped compiler-intermediate languages. We present a language that tracks the points in the program where data are defined or used, as part of the intermediate language's type system. We show how a simple transformation allows the compiler writer, using this language, to use multiple representations for data in the same program --- even when incompatible representations apparently flow to the same program point. We show results obtained from mixing different representations of functions in a compiler for Standard ML.