Lightweight Verification with Dependent Types

Aaron Stump

Computer Science and Engineering Dept.
Washington University in St. Louis

Abstract

Dependent types, studied for many years in Logic, have recently been gaining attention in Functional Programming Languages for expressing rich properties as types. A simple example is a type $\langle list\ A\ n\rangle$, for lists of length n holding objects of type A. A more complex example is $\langle trm\ G\ T\rangle$, for terms in some object language which have object-language type T in context G. Dependently typed programming languages seek to support static verification of code manipulating such data types, by statically enforcing the constraints the data types impose. The verification is lightweight in the sense that the aim is typically to verify preservation of datatype properties, rather than full functional specifications of programs.

This talk will explore dependently typed programming in the context of Guru, a new dependently typed programming language under development at Washington University in St. Louis. Guru lifts the restriction to terminating programs which is commonly required by dependently typed programming languages (such as Coq, Epigram, and ATS, to name just a few). This is done by the novel technical feature of strictly separating program terms from proofs, and types from formulas, thus going counter to the commonly used Curry-Howard isomorphism. We will consider dependently typed programming in Guru via several examples: tree operations which are statically verified to preserve the binary search tree property, and compilation of simply typed object programs which is statically verified to preserve the programs' object-language type.