Trustworthy decompilation: extracting models of machine code inside an ITP

Magnus O. Myreen

Computer Laboratory, University of Cambridge

Abstract

Modern processors support a large numbers of instructions and a multitude of features; as a result, detailed formal models of real instruction set architectures (ISAs) are long and hard to understand. Established approaches for proving functional properties on top of these models tie proofs to a specific model and require expert knowledge of the underlying model and substantial manual effort of those performing the proofs.

In this talk, I will explain a novel approach to verification of machine code which addresses these issues. My approach is based on translation: machine-code programs are translated into functionally equivalent tail-recursive functions via fully-automatic deduction. In doing so, the problem of proving properties of machine-code programs reduces to a problem of proving properties of recursive functions. My approach has several advantages over established approaches of verification condition generation. In particular, the new approach does not require annotating the program with assertions; and, more importantly, this approach separates the verification proof from the underlying ISA models so that specific resource names, some instruction orderings and certain control-flow structures become irrelevant. As a result, proof reuse is enabled to a greater extent than in established methods.

Towards the end of the talk, I will summarise some lessons that were learnt when implementing this tool for the HOL4 theorem prover. I will also explain some applications of this decompiler in automatic synthesis of correct code from functional specifications and explain how verification combined with synthesis has been used in case studies such as the construction of formally verified implementations of Lisp in ARM, x86 and PowerPC machine code.

The automation described above has been implemented as ML programs which steer HOL4 to a translation proof for each input machine-code program.