Towards the Next Generation of Proof Assistants: Enhancing the Proofs–as–Programs Paradigm

Liron Cohen

As software has grown increasingly critical to our society's infrastructure, mechanically-verified software has grown increasingly important, feasible, and prevalent. Proof assistants have seen tremendous growth in recent years because of their success in the mechanical verification of high-value applications in many areas, including cyber security, cyber-physical systems, operating systems, compilers, and microkernels. These proof assistants are built on top of constructive type theory whose computational interpretation is given by the proofs-as-programs paradigm, which establishes a correspondence between formal proofs and computer programs. However, while both proof theory and programming languages have evolved significantly over the past years, the cross-fertilization of the independent new developments in each of these fields has yet to be explored in the context of this paradigm. This naturally gives rise to the following questions: how can modern notions of computation influence and contribute to formal foundations, and how can modern reasoning techniques improve the way we design and reason about programs?

In this talk I first demonstrate how using programming principles that go beyond the standard lambda-calculus, namely state and non-determinism, promotes the specification and verification of modern systems, e.g. distributed systems. I then illustrate the surprising fragility of proof assistants in the presence of such new computational capabilities, and outline my ongoing efforts to develop a more robust foundation. For the converse direction, I show how incorporating modern proof-theoretic techniques offers a more congenial framework for reasoning about hard programming problems and hence facilitates the verification effort.