Rehabilitating CPS

Olin Shivers
MIT Artificial Intelligence Laboratory
shivers@ai.mit.edu

In the eighties, if one set out to write a compiler for a lambda-calculus based language, the odds are one would have chosen an intermediate representation based on continuation-passing style—"CPS." Compilers such as Orbit, Kelsey's transformational compiler, and SML/NJ all traced their heritage back to the seminal influence of Steele's Rabbit compiler, which established the CPS-as-intermediate-representation thesis. While CPS by no means had an exclusive franchise, it staked out a large niche in the functional-language arena—at many institutions, it was simply the accepted and expected representational framework.

However, at the turn of the decade, researchers began increasingly to experiment with alternative low-level, lambda-based frameworks. Representations such as nqCPS and A-normal form allowed compiler writers to factor the features provided by CPS, such as serialisation of primitive operations, and the naming of all intermediate results, without having to commit to the explicit machinery of exposed continuations. The new and interesting compilers, such as Morrisett and Tarditi's TIL compiler, were written using "direct style" lambda-calculus intermediate representations. CPS has faded from view somewhat; I am not aware of a serious CPS-based compiler being written in the last five to eight years.

Part of the reason academe's attention turned away from CPS was a series of influential papers by Felleisen and Sabry establishing some deep formal connections between the analytic power of the two frameworks. The message of these papers was that introducing explicit continuations provided no extra theoretical benefit—so why bother with them?

As another decade prepares to turn over, we have the advantage of ten years of experience working with these alternative frameworks. I suggest that it is now time to reexamine the benefits of CPS, in the light of these lessons learned.

In this talk, I will discuss the benefits of CPS from a modern perspective. I'll show some interesting examples of recent work on operating-system concurrency and transducer composition that rely critically on the ability to represent continuations explicitly. I will discuss how we can apply the lessons of the last decade to CPS, and *vice versa.* I'll address the issue of formal equivalences between continuation-based and direct-style representations, and point out limits in our current understanding and use of continuation-based compiler frameworks.