JACQUES CARETTE, McMaster University, Hamilton, Canada

Functors, CPS and monads, or how to generate efficient algebraic code from abstract designs

Using monads and Ocaml's advanced module system in a generative context, it is possible to totally eliminate the overhead of abstraction. This lets one use extreme forms of *information hiding* at no run-time cost. Furthermore the typed nature of the generative context provide static guarantees about the generated code. The various aspects of the algorithms can be made completely orthogonal and compositional, even in the presence of name-generation for temporaries and other bindings as well as "interleaving" of aspects. We also show how to encode some domain-specific knowledge so that "clearly wrong" compositions can be statically rejected by the compiler. All our examples are drawn from numerical and symbolic linear algebra (Gaussian Elimination, LU Decomposition, *etc.*).

Joint work with Oleg Kiselyov.