More Prehistory of Programming Languages

Matthias Felleisen

Lambda Calculus Models of Programming Languages

@phdthesis{m:lc-models,
  author = {James H. Morris},
  title = "Lambda-Calculus Models of Programming Languages",
  school = "Massachusetts Institute of Technology",
  year = 1968
}

Contents

Morris investigates the “semantics” of two distinct aspects of programming languages: recursion and types. In the course of investigating recursion, he invents operational equivalence aka observational equivalence.

Meta-Circular Definitions

@inproceedings{jr:definitional,
  author = {John C. Reynolds},
  title = "Definitional interpreters for higher-order programming languages",
  booktitle = "Proceedings of the ACM Annual Conference",
  year = 1972,
  pages = {717--740}
}

Contents

Reynolds explains the limitations of meta-interpreters. For example, meta-interpreters don’t explain the order-of-evaluation for function application in a higher-order languages. They also fail to bring across what control constructs or mutation really mean. He demonstrates that by exploiting higher-order constructs one can improve the meta-circular interpreters. To explain the order of evaluation, he invents continuation-passing style; for mutation, he introduces a store.