CMPT 383
FP for Compilation

Anders Miltner
Overview

- Why are functional languages used for compilation
- Pros and Cons
- Interpretation vs Compilation
- Verified compilation
- Types
Warning — Interactive Lecture

• I’ll be live coding

• Stop me if I’m going too fast

• I’ll post the written code at the end of the lecture
Functional Languages for Compilation

- Compilation is a pure mathematical process
- Languages can be modelled well using FP-style ADTs

Note: No side effects necessary!
General Compilation Pipeline

1. Source Language Text
2. Parsing
3. Source Language AST
4. Compilation
5. Target Language AST
6. Pretty Printing
7. Target Language Text
GHC Compilation Pipeline

Haskell Language Text
  Parsing
  Haskell Language AST
  Compilation
  x86 Language AST
  Pretty Printing
  x86 Language Text
Pros and Cons

Pros

• Clear description of source and target language
• “Readable code”
• Fits well with functional style

Cons

• Slower compilation times
Toy Languages — Arith and Stack

Source Language

• Simple language of numeric operations
• Numbers, plus, minus, mult
• More, if we get to well-typedness

Target Language

• Stack-based evaluator
• We have an instruction set and a stack of numbers, and nothing else

Taken from: https://softwarefoundations.cis.upenn.edu/qc-current/toc.html
Programming Time
What does a language *mean*?

• Just a way to tell a computer what to do?

• Is the meaning of a language defined by the compiler?
  
  • What if the compiler has a bug?

• How do you interpret compilation to different architectures

• Some sort of deeper mathematical idea?

• Perhaps we can give an explicit “semantics” to the language
Interpreters as Semantics

- Write a clean formalism describing how a program computes
- This is essentially an interpreter!
- Caveat: Real interpreters care about speed, semantics care about mathematical cleanliness
Programming Time
Verified Compilation

• Relate a compiler and two formal semantics

• Ensure that the following diagram commutes:

• William Bowman at UBC works on this
Types

• “Well-Typed programs don’t go wrong”

• You can ensure that programs that satisfy a given typing have a well-defined semantics

• You can ensure that programs that satisfy a given typing
Programming Time