Principles of Programming Languages

Introduction to Haskell

What is Haskell?

- Haskell is a lazy, functional programming language created in the late 1980s by a committee of academics.
- There were a plethora of lazy functional languages around, everyone had their favorite, and it was hard to communicate ideas.
- So, a bunch of people got together and designed a new language, taking some of the best ideas from existing languages (and a few new ideas of their own). Haskell was born.

What is Haskell?

a typed, lazy, purely functional language

Haskell is functional

There is no precise, accepted meaning for the term “functional”. But when we say that Haskell is a functional language, we usually have in mind two things:

1. Functions are first-class, that is, functions are values which can be used in the same ways as any other sort of value.
2. The meaning of Haskell programs is centered around evaluating expressions rather than executing instructions.

Haskell is statically typed

- Everything has a type
- Everything .must make sense at compile time

Haskell is pure

Haskell expressions are always referentially transparent, that is:

- No mutation! Everything (variables, data structures...) is immutable.
- Expressions never have "side effects" (like updating global variables or printing to the screen).
- Calling the same function with the same arguments results in the same output every time.

Haskell is lazy

- You don't evaluate an expression until its result is absolutely necessary

- Haskell's evaluation strategy is called call-by-need
  - Because of the other properties: you actually only evaluate an expression once and cache the result

Why is this cool?

- Algebraic laws: equational reasoning & optimizations
  - Can always replace things that are equal, λ calculus!
- Easier to think about
  - e.g., don't need to worry if x changed after calling f
- Parallelism
  - Can evaluate expressions in parallel!

Why is this cool?

- Can define your own control structures using functions
  - E.g., defining if-then-else is much easier in Haskell can be done naturally

- Can define infinite data structures
  - E.g., infinite lists, trees, etc.
  - Can solve general problem and then project solution

NEXT – Intro to GHC