Programming Language Concepts Notes

Marie Burer

18-1-24

Contents

1 Introduction to Haskell

Functional programming: Functions are central

Garbage collection came first with FP, and recursion is used over loops.

Avoid side effects

Side effects include changing some hidden state. Functions should return the same values when given the same input ideally.

Haskell enforces purity, no side effects

Haskell is statically typed, lazy semantics

FP also used to make compilers and interpreters

Learning Haskell is good for you, a new paradigm and way to think

2 Higher-order functions in Haskell

In FP, functions are first-class

- Can be outputs
- Can be inputs

To call a function f with argument a, f a

(f . g) x = f (g x)

Consider:

f :: (a → b) → (a,a) → (b,b)

f g (x,y) = (g x, g y)

Anonymous function:

$\backslash x \rightarrow x * x$

In definitions:

$id = \backslash x \rightarrow x$

As arguments to higher order functions:

applyTwice($\backslash x \rightarrow x * x$)

2.1 Partial Applications and Sections

Suppose we have f :: Int → Int → Int

Then we can write f 3 :: Int → Int

Int → Int → Int is the same as Int → (Int → Int)

If we have something like (+10), that is a section

2.2 List Combinators

Consider map :: (a → b) → ([a] → [b])

f :: a → b

map f xs applies f to all elements of xs

filter :: (a → Bool) → [a] → [a]

filter p xs keeps every element of xs where p x is True

zipWith :: (a → b → c) → [a] → [b] → [c]

zipWith f [x1, ..., xn][y1, ..., y(n+k)] = [f x1 y1, ..., f xn yn]

foldr f z xs = f x1 (f x2 (... (f xn z)))

3 Data Types and Recursion

data BinTree = Leaf | Node Char BinTree BinTree

We can write a function on this data type like

height :: BinTree a $\rightarrow$ Int

height Leaf = 0

height (Node x t1 t2) = 1 + max (height t1) (height t2)

4 Stuff I didn't take notes on

5 Introduction to Agda

How to know what is true?

In Agda, formulas are expressed as types.

The equation $1 + 1 = 2$ becomes a type

When type-checking, test if two terms are definitionally equal