CS350 2024-25 Sem I

Satyadev Nandakumar

August 7, 2024

Outline

Lecture 3 contents

defining and applying simple functions

Multi-argument functions

Standard Prelude

The library of standard functions.

1. Open `ghci`
2. Prelude.hs is automatically loaded.
3. type `:browse` to see all functions loaded from Prelude
4. Try `:info take`, `:doc take`, and `:type take`

Function application

- If `foo` is a function with 2 arguments, you can apply the function as

`foo x y` where `x` and `y` are arguments.

- Function application has higher priority than any other operation.

This means

```
f a + b
```

is

```
(f a)+b
```

- Question: Is `f g x` the same as `f (g x)`?

Function application

- If `foo` is a function with 2 arguments, you can apply the function as

`foo x y` where `x` and `y` are arguments.

- Function application has higher priority than any other operation.

This means

```
f a + b
```

is

```
(f a)+b
```

- Question: Is `f g x` the same as `f (g x)`?

(No!, the first is taken as 2 arguments to a 2 argument function.)

Writing haskell scripts

- Comments : lines starting with `--`
- Save your file as <filename>.hs
- Either start `ghci` and type `load <filename>.hs`
- Or, if your code has a `main`, then compile it as `ghc <filename>.hs`
- In `ghci`, after any change in file, you can type `:reload`

Functions on basic types

`plus`

```
plus a b = a+b
```

Functions on basic types

plus

```
plus a b = a+b
```

isPositive

The following code uses guards

```
isPositive n | n<= 0      = False
             | otherwise  = True
```

Functions on basic types

plus

```
plus a b = a+b
```

isPositive

The following code uses guards

```
isPositive n | n<= 0      = False
             | otherwise  = True
```

We can also use conditionals

```
isPositive2 n = if n<=0 then False else True
```

Lambda expressions

Functions can have no name (anonymous)

Example:

```
(\ x -> x+1)
```

is an anonymous function.

```
(\ x -> x+1) 2
```

evaluates to 3.
The \ is called lambda, since it reminds us of λ, the binding symbol used in λ calculus, after which Haskell is modeled.

Functions with list arguments

Pattern-matching

Length of a list

Reversing a list

Polymorphism

Try equality on lists!

Length

What is polymorphism?

Look at the type of the function. If it has a type variable, then the function is polymorphic.

Polymorphism

Try equality on lists!

Length

What is polymorphism?

Look at the type of the function. If it has a type variable, then the function is polymorphic.

```
:t len
```

outputs

```
Num p :: [a] -> p
```

a and p are type variables. The function maps list of elements of any type (represented by the type variable a) to a number.

Higher-order programming with functions

Map

Filter

Fold

Understand the composition function in base

- Type :browse
- Find the function (.)
- Understand its type
- From its type we can infer its function

Where to find help

- Type :t to find the type of a function
- Type :i to find basic information
- Type :browse <module> to find list of functions in the module
- Type :doc

A problem to think about

A game: given a word like "food", see whether you convert it into another word, like "soul", using five transition words, where each word differs from the immediately previous one by exactly one word. For example, changing "Foot" into "Ball"

1. Foot
2. Food
3. Fold
4. Bold
5. Bald
6. Ball

Write a Haskell program, that, given two words each having length 4, outputs a sequence of five transition words if a sequence exists, and otherwise outputs the empty list.