Lecture 15: The Lambda Calculus

Dr John Levine

CS103 Machines, Languages and Computation
November 23rd 2015

Lecture and Tutorial Plan

- Mon 23 Nov: introducing the lambda calculus
- Homework: Assignment 9 in the workbook
- Thu 26 Nov: tutorial on Assignments 8 and 9
- Fri 27 Nov: NO LECTURE

- Mon 30 Nov: writing recursive Python functions
- Homework: Assignment 10 in the workbook
- Thu 3 Dec: tutorial on Assignment 10, general revision
- Fri 4 Dec: Class Test 2, 10am, Assembly Hall

The Lambda Calculus

- What is the lambda calculus then?

  “A formal system for function definition, function application and recursion.”

- Invented by Alonzo Church in the 1930s as part of an investigation into the foundations of mathematics
- Now used as a device in the theory of programming languages and computation (amongst others)
- Can be used to write programs and was the inspiration for functional programming languages (Lisp, Haskell)

Simple Function Application

- We apply a function to an argument like this:

  function argument

- For example:

  even 4
  not True
  addthree 5
  length “fred”
  integer “abc”
  identity 456
  identity “fred”

University of Strathclyde Science

Simple Function Application

- Functions return results:

  function argument → result

- For example:

  even 4 → True
  not True → False
  addthree 5 → 8
  length “fred” → 4
  integer “abc” → False
  identity 456 → 456
  identity “fred” → “fred”

Simple Function Application

- Functions have types:

  function (type → type) argument → result

- For example:

  even (int → bool) 4 → True
  not (bool → bool) True → False
  addthree (int → int) 5 → 8
  length (string → int) “fred” → 4
  integer (α → bool) “abc” → False
  identity (α → α) 456 → 456
  identity (α → α) “fred” → “fred”

The Basic Lambda Calculus

- We use the lambda calculus to construct anonymous functions of a single variable:

  $\lambda x.x+3$

- This is a function of a single *bound variable* (x)
- The *body* of the function is x+3
- When we apply this function to any integer n, it will return the value n+3:

  $(\lambda x.x+3)\ 2 \rightarrow 5$

- How does it do this?

Beta Reduction

- To find the result of applying a lambda expression to an argument, we use beta reduction:

  $(\lambda x.\mathit{body})$ a => *body* with x replaced by a

- Examples:

  $(\lambda x.x+2)$ 3
  $(\lambda y.y+y)$ 5
  $(\lambda s.\text{length } s)$ “abc”
  $(\lambda k.k)$ “fred”
  $(\lambda x.\lambda y.x+y)$ 2

Beta Reduction

- To find the result of applying a lambda expression to an argument, we use beta reduction:

  (λx.*body*) a => *body* with x replaced by a

- Examples:

  (λx.x+2) 3 => 3+2 → 5
  (λy.y+y) 5 => 5+5 → 10
  (λs.length s) “abc” => length “abc” → 3
  (λk.k) “fred” => “fred”
  (λx.λy.x+y) 2 => λy.2+y

Lambda Expressions Have Types

- Lambda expressions are functions of a single variable
- They therefore have a type, just like simple functions of a single variable:

$(\lambda x.x+2)\ (\text{int} \rightarrow \text{int})$
$(\lambda y.y+y)\ (\text{int} \rightarrow \text{int})$
$(\lambda s.\text{length } s)\ (\text{string} \rightarrow \text{int})$
$(\lambda k.k)\ (\alpha \rightarrow \alpha)$
$(\lambda x.\lambda y.x+y)\ (\text{int} \rightarrow (\text{int} \rightarrow \text{int}))$

Functions with Two Arguments

- What if we want to write a function of two variables?

  F(x,y) = sqrt (x*x + y*y)

- We have to pass in the arguments one at a time:

  (λx.λy.sqrt (x*x + y*y)) 3 4

- Apply the lambda expression to the arguments from left to right:

  => (λy.sqrt (3*3 + y*y)) 4
  => sqrt (3*3 + 4*4))
  → 5

Functions as Arguments

- Sometimes, functions are arguments to other functions
- Example: mapcar in Lisp takes two arguments: a list of items and a function to apply to those items
- We can write lambda expressions which bind functions and then apply these to other lambda expressions:

$(\lambda f.f\ 3)\ (\lambda x.x+2)$

$\Rightarrow (\lambda x.x+2)\ 3$

$\Rightarrow 3+2$

$\rightarrow 5$

Assignment 9

1. Write the following functions as lambda expressions:

   (a) f(x) = x+5
   (b) f(v) = and(v,v)
   (c) f(x,y) = x+y
   (d) f(x,y,s) = (x+y)*(length s)

2. Perform beta reduction on the following:

   (a) (λx.even x) 2
   (b) (λs.(length s)+3) “abc”
   (c) (λa.λb.(length a)+(length b)) “fred” “eric”
   (d) (λv.λw.and(v,or(v,w))) True False

Assignment 9

3. Give the types of the lambda expressions in Q1 and Q2.

4. (hard) Consider the function application as given on the slide entitled “Functions as Arguments”:

   $(\lambda f.f\ 3)\ (\lambda x.x+2)$

   What is the type of the expression $(\lambda f.f\ 3)$?

Lecture and Tutorial Plan

- Mon 23 Nov: introducing the lambda calculus ✓
- Homework: Assignment 9 in the workbook
- Thu 26 Nov: tutorial on Assignments 8 and 9
- Fri 27 Nov: NO LECTURE

- Mon 30 Nov: writing recursive Python functions
- Homework: Assignment 10 in the workbook
- Thu 3 Dec: tutorial on Assignment 10, general revision
- Fri 4 Dec: Class Test 2, 10am, Assembly Hall