Functional Programming for BDA - List 2

Maps, folds and lambda expressions

Marcin Michalski, DCS FFPT WUST 2019/2020

Treat exercises as a warm up. Submit tasks 1-4 only.

Exercise 1. Implement functions

a) $a(x, y) = x + y$;

b) $b(x, y) = \text{const}_x(y) = x$;

c) $\pi_2(x, y) = y$;

d) $\text{id}(x) = x$;

using lambda calculus.

Exercise 2. Simplify the following expression:

$$(\backslash x \to \backslash y \to x + 2 * y)(x + 2 * y).$$

Tip: Rename colliding variables (it's called α-conversion) and apply $(\backslash x \to \backslash y \to x+2*y)$ to the expression $(x + 2 * y)$ (it's called β-reduction).

Exercise 3. Let f and g be functions. Express `f.g` (the composition of f and g) in the lambda calculus. What is the type of `(.)` as a function?

Exercise 4. Let `p1 = \x y -> x` and `p2 = \x y -> y`. Calculate `p1.p2` and `p2.p1`.

Exercise 5. Calculate the output of `foldl (^) 1 [2,2,2]` and `foldr (^) 1 [2,2,2]`.

Exercise 6. Calculate the output of `(head $map (\x y -> x+2*y) [7,11,13]) 5`.

Task 1. Express `map` via `foldr`. Use lambda expressions.

Task 2. Implement a function that for a given list of integers returns the sum of squares of its even members. Use fold.

Task 3. Implement a function that for a given list of natural numbers calculates how many members of the list are prime. It should be reasonably fast. Use fold.

Task 4. Implement a function that for a given natural number n calculates the approximation of e, i.e. $\sum_{k=0}^{n} \frac{1}{k!}$. It should do so fast - in a linear time.