Programação Funcional (MCCC015-23)
Lista de Exercícios 1 - Cálculo λ

Emilio Francesquini
e.francesquini@ufabc.edu.br
Universidade Federal do ABC

5 de julho de 2024

Em alguns exercícios nesta lista utilizamos a sintaxe de Haskell para expressões em cálculo λ, incluindo os açúcares sintáticos. Veja a tabela abaixo para exemplos de conversões entre as notações:

Cálculo λ	Haskell
x	`x`
$\lambda x\ .\ \ldots$	`\x -> ...`
$\lambda x\ .\ \lambda y\ .\ \lambda z\ .\ \ldots$	`\x -> \y -> \z -> ...`
$\lambda x\ .\ \lambda y\ .\ \lambda z\ .\ \ldots$	`\x y z -> ...`
$(\lambda x\ .\ x)\ y$	`(\x -> x) y`

1. Adicione parênteses às expressões abaixo para evidenciar a ordem de avaliação:

   (a) $\lambda x\ .\ x\ \lambda y\ .\ y\ x$

   (b) $(\lambda x\ .\ x)\ (\lambda y\ .\ y)\ \lambda x\ .\ x\ (\lambda y\ .\ y)\ z$

   (c) $(\lambda f\ .\ \lambda y\ .\ \lambda z\ .\ f\ z\ y\ z)\ p x$

   (d) $\lambda x\ .\ x\ \lambda y\ .\ y\ \lambda z\ .\ z \lambda w\ .\ w\ z\ y\ x$

2. Reescreva as expressões do Exercício 1 utilizando a sintaxe de Haskell.

3. Considere a função `DUAS_VEZES = \f x -> f (f x)` e a função `INC` dada em aula. Utilizando-as, defina as funções:

   (a) `SOMA_QUATRO` que devolve o número passado como argumento somado a 4

(b) `SOMA_DEZESSEIS` que devolve o número passado como argumento somado a 16

(c) `SOMA_SESSENTA_E_QUATRO` que devolve o número passado como argumento somado a 64

4. Utilize reduções β (e α caso ache conveniente) para reduzir as expressões abaixo para a sua forma normal (caso ela exista).

   (a) `(\ g -> g 5) (\ x -> add x 3)`

   (b) `(\x -> (\ y z -> z y) x) p (\x -> x)`

   (c) `(\x -> x x x) (\x -> x x x)`

   (d) `(\x -> \y -> (add x ((\x -> sub x 3) y))) 5 6`

   (e) `(\c -> c (\a -> \b -> b)) ((\a -> \b -> \f -> f a b) p q)`

   (f) `DUAS_VEZES (\n -> (mul 2 (add n 1))) 5`

   (g) `DUAS_VEZES (DUAS_VEZES (\n -> (mul 2 (add n 1)))) 5`

   (h) `DUAS_VEZES DUAS_VEZES sqr 2`

   (i) `(\x -> ((\z -> (add x x)) ((\x -> \z -> (z 13 x)) 0 div))) ((\x -> (x 5)) sqr)`

5. Utilizandos as funções `TRIPLE` e `TRD3` vistas em aula, mostre passo-a-passo que `TRD3 (TRIPLE p q r) = r`.

6. O cálculo λ como descrito em aula, admite funções com apenas um único parâmetro formal. Contudo muitas funções necessitam de um ou mais parâmetros (ex. soma, divisão, ...). Para lidar com esses casos temos duas abordagens possíveis: (i) utilizar pares (ou no caso geral, tuplas de tamanho apropriado) de elementos como argumentos; (ii) utilizar a versão **curried** da função que tem a propriedade de que seus argumentos são fornecidos um por vez. Por exemplo, a expressão `3 + 5` poderia ser escrita como `((ADD 3) 5)`. Considerando as funções `PAIR`, `FST` e `SND` dadas em aula e as funções abaixo:

   - `CURRY = \f x y -> f (PAIR x y)`
   - `UNCURRY = \f p -> f (FST p) (SND p)`

   Mostre que:

   (a) `CURRY (UNCURRY h) = h`

(b) `UNCURRY (CURRY h) (PAIR r s) = h (PAIR r s)`

7. Diversas linguagens de programação, em particular as funcionais, fornecem a construção `let` para criar e inicializar uma variável com um valor. Converta as expressões que usam `let` abaixo para expressões que usam apelas lambdas apenas.

   (a) `let x = 5 in let y = (add x 3) in (mul x y)`

   (b) `let a = 7 in let g =  x . (mul a x) in let a = 2 in (g 10)`

8. Forneça a forma normal das expressões do exercício 7.

9. Baseando-se na implementação de `INC` dado em aula e na definição dos números de Church, forneça a implementação da operação `MULT = \a b -> ...` que recebe dois números inteiros `a` e `b` e devolve o valor de `a` multiplicado por `b`.

10. Utilizando o combinador Y, implemente a função recursiva `FIB` que dado `n` devolve o n-ésimo número da sequência de Fibonacci. Exemplos:

    - `Y FIB ZERO` resulta em `ZERO`
    - `Y FIB DOIS` resulta em `UM`
    - `Y FIB SEIS` resulta em `OITO`

11. Utilizando cálculo λ, as funções dadas em aula e as funções desenvolvidas nos exercícios anteriores, implemente uma função `POT` que recebe 2 números inteiros, `b` e `p` e que devolva o valor b^p.