**Lista 3 de LP**

Douglas Rodrigues de Almeida

1. Determine o tipo de *λx. λy. λz. xz(yz)*, usando a técnica-informal-de-inferência-de-tipos.

O tipo da expressão é da forma:

tx → ty → tz → tr

onde:

tx representa o tipo de x.

ty representa o tipo de y.

tz representa o tipo de z.

tr. representa o tipo de xz(yz)

A partir da expressão xz(yz), concluímos:

* y tem tipo funcional pois recebe z como e entrada e retorna algo na saída. Logo, é da forma t1 → t2, onde t1 = tz.
* x também tem tipo funcional. Ele recebe z e a saída de y como entrada e retorna algo na saída. Logo, é do tipo t3 → t4 → t5, onde t3 = tz e t4 = t2.
* O tipo do resultado de xz(yz) é tr. Mas, tr = t5 pois é o tipo da saída de x.

Usando estas informações, obtemos o tipo de λ*x*. λ*y*. λ*z*. *xz*(*yz*) como:

tx → ty → tz → tr

(tz → t2 → t5) → (tz → t2) → tz → t5

que pode ser reescrito como:

(a → b → c) → (a → b) → a → c

1. Use a técnica-informal-de-inferência-de-tipos para determinar o tipo da função either definida abaixo:

Sendo o tipo *Either* definido como:

data Either a b = Left a | Right b

Defina como é inferido o tipo de either, usando a técnica-informal-de-inferência-de-tipos, definido como:

either f g (Left x) = f x  
 either f g (Right y) = g y

O tipo da expressão é da forma:

ta → tb → te → tr

onde:

ta representa o tipo de f.

tb representa o tipo de g.

te representa o tipo de data Either a b.

tr. representa o tipo de Left a ou de Right b

A partir da expressão data Either a b = Left a | Right b, concluímos:

* a tem tipo funcional pois recebe x como e entrada e retorna algo na saída. Logo, é da forma t1 → t2.
* b também tem tipo funcional. Ele recebe y e retorna algo na saída. Logo, é do tipo t3 → t4.
* e é o tipo Either a b, ou é o tipo a ou é o tipo b, assim é ou t1 ou t3.
* O tipo do resultado de Left a ou de Right b é tr. Mas, tr é o do tipo t2 ou t4 pois são as saídas de f ou de g. Logo tr = t2 = t4.

Assim:

ta → tb → te → tr

(t1 → t2) → (t3 → t2) → Either t1 t3  → t2

que pode ser reescrito como:

(a → c) → (b → c) → Either a b → c

1. Defina expressão ou função com tipo:
2. (*Ord* a, *Show* a) => a -> a -> *String*

exibirMenor :: (Ord a, Show a) => a -> a -> String

exibirMenor a b

| a < b = show a

| otherwise = show b

1. (a -> b -> c) -> b -> a -> c

flip :: (a -> b -> c) -> b -> a -> c  
flip f x y = f y x

1. (*a* -> *b*, *a* -> *c*) -> *a* -> (*b*, *c*)

funcao :: (a -> b, a -> c) -> a -> (b, c)

funcao (f, g) x = (f x, g x)

1. (*a* -> c, *b* -> *d*) -> (*a*, *b*) -> (*c*, *d*)

funcao :: (a -> c, b -> d) -> (a, b) -> (c, d)

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

1. (*b* -> *c*) -> (*a*->*b*) -> *a* -> *c*

(.) :: (b -> c) -> (a -> b) -> a -> c  
f . g = \ x -> f (g x)

1. [(*a*, *b*)] -> ([*a*], [*b*])

unzip :: [(a, b)] -> ([a], [b])  
unzip = foldr (\(a,b) ~(as,bs) -> (a:as,b:bs)) ([],[])

1. (*a* -> *b* -> *a*) -> *a* -> [*b*] -> [*a*]

scanl         :: (a -> b -> a) -> a -> [b] -> [a]  
scanl f q xs  =  q : (case xs of  
                            []   -> []  
                            x:xs -> scanl f (f q x) xs)