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:

   $$t_x \rightarrow t_y \rightarrow t_z \rightarrow t_r$$

   onde:

   t_x representa o tipo de x.
   t_y representa o tipo de y.
   t_z representa o tipo de z.
   t_r. 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 $t_1 \rightarrow t_2$, onde $t_1 = t_z$.
   - x também tem tipo funcional. Ele recebe z e a saída de y como entrada e retorna algo na saída. Logo, é do tipo $t_3 \rightarrow t_4 \rightarrow t_5$, onde $t_3 = t_z$ e $t_4 = t_2$.
   - O tipo do resultado de xz(yz) é t_r. Mas, $t_r = t_5$ pois é o tipo da saída de x.

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

   $$t_x \rightarrow t_y \rightarrow t_z \rightarrow t_r$$
   $$(t_z \rightarrow t_2 \rightarrow t_5) \rightarrow (t_z \rightarrow t_2) \rightarrow t_z \rightarrow t_5$$

   que pode ser reescrito como:

   $$(a \rightarrow b \rightarrow c) \rightarrow (a \rightarrow b) \rightarrow a \rightarrow c$$

2. 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:

   $$t_a \rightarrow t_b \rightarrow t_e \rightarrow t_r$$

   onde:

   t_a representa o tipo de f.
   t_b representa o tipo de g.
   t_e representa o tipo de data Either a b.
   t_r. 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 $t_1 \rightarrow t_2$.

- b também tem tipo funcional. Ele recebe y e retorna algo na saída. Logo, é do tipo $t_3 \rightarrow t_4$.
- 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 t_r. Mas, t_r é o do tipo t_2 ou t_4 pois são as saídas de f ou de g. Logo $tr = t_2 = t_4$.

Assim:

$$t_a \rightarrow t_b \rightarrow t_e \rightarrow t_r$$
$$(t_1 \rightarrow t_2) \rightarrow (t_3 \rightarrow t_2) \rightarrow \text{Either } t_1\ t_3 \rightarrow t_2$$
que pode ser reescrito como:
$$(a \rightarrow c) \rightarrow (b \rightarrow c) \rightarrow \text{Either a b} \rightarrow c$$

3. Defina expressão ou função com tipo:

a. (*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
```

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

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

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

```
funcao :: (a -> b, a -> c) -> a -> (b, c)
funcao (f, g) x = (f x, g x)
```

d. (*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)
```

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

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

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

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

g. (*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)
```