


Paradigma Funcional

Tópicos Adicionais



ERICK GALANI MAZIERO
erick.maziero@ufla.br

Departamento de Ciências da Computação
Universidade Federal da Lavras



Encadeamento de funções (currying)

Currying é uma técnica de transformação de uma função que recebe múltiplos parâmetros, de forma que ela pode ser chamada como uma cadeia de funções que recebem somente um parâmetro cada.

$$\begin{aligned} f &: (X \times Y) \rightarrow Z \\ \text{curry}(f) &: X \rightarrow (Y \rightarrow Z) \end{aligned}$$

f toma um parâmetro do tipo X e retorna uma função do tipo $Y \rightarrow Z$





Encadeamento de funções (currying)

```
soma :: Int -> Int -> Int  
soma x y = x + y
```

```
inc :: Int -> Int  
inc = soma 1
```

Execução:

```
Prelude> soma 4 5
```

```
9
```

```
Prelude> inc 4
```

```
5
```





Encadeamento de funções (currying)

```
div x y = x / y
```

```
inv = div 1
```

Execução:

```
Prelude> div 4 5
```

```
0.8
```

```
Prelude> inv 4
```

```
0.25
```





Avaliação Preguiçosa - i

Várias linguagens funcionais, incluindo Haskell, utilizam-se de uma técnica denominada de avaliação das funções denominada avaliação preguiçosa (*lazy evaluation*).

A técnica consiste em não avaliar nenhuma subexpressão ou função até que seu valor seja reconhecido como necessário.



Avaliação Preguiçosa - ii

Suponha a seguinte função em Haskell:

```
square x = x * x
```

Seja a avaliação de:

```
square ( square ( square 2 ) )
```

Avaliação Preguiçosa - ii

Com avaliação tradicional:

```
⇒ square (square (square 2))  
⇒ (square (square 2)) * (square (square 2))  
⇒ ((square 2) * (square 2)) * (square (square 2))  
⇒ ((2 * 2) * (square 2)) * (square (square 2))  
⇒ (4 * (square 2)) * (square (square 2))  
⇒ (4 * (2 * 2)) * (square (square 2))  
⇒ (4 * 4) * (square (square 2))  
⇒ 16 * (square (square 2))  
⇒ ... ⇒ 256
```

Avaliação Preguiçosa - iii

Com avaliação ávida:

⇒ square (square (square 2))
⇒ square (square (2 * 2))
⇒ square (square 4)
⇒ square (4 * 4)
⇒ square 16
⇒ 16 * 16
⇒ 256

Com avaliação preguiçosa:

⇒ square (square (square 2))
⇒ (square (square 2)) * (square (square 2))
⇒ ((square 2) * (square 2)) * ((square 2) * (square 2))
⇒ ((2 * 2) * (2 * 2)) * ((2 * 2) * (2 * 2))
⇒ (4 * 4) * (4 * 4)
⇒ 16 * 16
⇒ 256



Avaliação Preguiçosa - Outro Exemplo

Suponha:

```
menor x y  
  | x < y = x  
  | otherwise = y
```

```
dobro x = x + x
```

```
triplo x = 3 * x
```

```
f a b = (dobro (triplo (menor a b)))
```





Avaliação Preguiçosa - Outro Exemplo

Ao executarmos $f(2, 3)$, o cálculo é dado por:

```
f 2 3 = (dobro (triplo (menor 2 3)))  
      = (triplo (menor 2 3)) + (triplo (menor 2 3))  
      = (3 * (menor 2 3)) + (triplo (menor 2 3))  
      = (3 * (menor 2 3)) + (3 * (menor 2 3))  
      = (3 * 2) + (3 * (menor 2 3))  
      = (3 * 2) + (3 * 2)  
      = 6 + (3 * 2)  
      = 6 + 6  
      = 12
```





Referência Bibliográfica

Sebesta, R. W. (2011). *Conceitos de Linguagens de Programação*. 9 ed. Bookman.

Capítulo 15

<http://learnyouahaskell.com/chapters>

<https://www.tutorialspoint.com/haskell/index.htm>

