PROGRAMAÇÃO FUNCIONAL: Tipos de Dados Algébricos

Professor Rafael Kingeski

Departamento de Ciência da Computação
Centro de Ciências Tecnológicas - CCT
UDESC - Joinville.

Índice

Índice

Tipos de Dados Algébricos

Até agora, encontramos muitos tipos de dados. Bool , Int , Char, etc. Mas como fazemos o nosso próprio? Bem, uma maneira é usar a palavra-chave data para definir um tipo. Vamos ver como o tipo Bool é definido na biblioteca padrão.

```
data Bool = False | True
```

`data` significa que estamos definindo um novo tipo de dados.
A parte antes do = denota o tipo, que é Bool . As partes após o = são construtores de valor . Eles especificam os diferentes valores que esse tipo pode ter. O | é lido como ou . Portanto, podemos ler isso como: o tipo Bool pode ter um valor True ou False . Tanto o nome do tipo quanto os construtores de valor devem ser maiúsculos.

Tipos de Dados Algébricos

E se nós criarmos um "data" definido da mesma forma que os dados já existentes na linguagem Haskell???

```
data Shape = Circle Float Float Float | Rectangle
Float Float Float Float
```

E se definirmos uma função com entrada Shape e saída Float para calcular a área das formas geométricas definidas anteriormente?

```
surface ::  Shape -> Float
surface (Circle _ _ r) = pi * r ^ 2
surface (Rectangle x1 y1 x2 y2) = (abs $ x2 - x1) *
(abs $ y2 - y1)
```

Tipos de Dados Algébricos

Tente agora atribuir valores a um círculo.

```
ghci> Circle 10 20 5
```

E se aplicarmos uma função de ordem superior para preencher os raios de diferentes círculos em um mesmo ponto?

```
ghci> map (Circle 10 20) [4,5,6,6]
```

Sinônimos de Tipos

Os sinônimos de tipo não fazem nada em si, eles apenas dão nomes diferentes a alguns tipos para que façam mais sentido para alguém que lê nosso código e documentação.
Por exemplo como a biblioteca padrão define String como sinônimo de [Char].

Para criar um sinônimo de tipo utilizamos a palavra-chave "type".
A palavra-chave pode ser enganosa para alguns, porque na verdade não estamos criando nada novo (fizemos isso com a palavra-chave data), mas estamos apenas criando um sinônimo para um tipo já existente.

```
type Agenda = [(String,String)]
```

Sinônimos de Tipos


```
type Numerodetelefone = String
type Nome = String
type Agenda = [(Nome,Numerodetelefone)]
```

Índice

IO em Haskell

Olá, mundo!

```
main ::  IO ()
main = putStrLn "olá, mundo!"
```

Para que o olá, mundo apareça na tela podemos compilar ou interpretar o programa criado.

Referências


Lipovaca, M.
Learn You a Haskell for Great Good!
(No Starch Press, 2011)