Universidade Federal de Goiás
INF - Instituto de Informática

Disciplina: Tópicos - Programação Funcional — Professor: Daniel Ventura
Curso: Ciências da Computação — Data: 03/10/2024

Laboratório 6: **Funções de Ordem Superior**

1. Usando `foldl`, defina uma função `dec2int` :: $[Int] \to Int$ que converte uma lista de dígitos decimais num inteiro. Exemplo: `dec2int` $[2,3,4,5] = 2345$.

2. As funções `foldl1` e `foldr1` do prelúdio-padrão são variantes de `foldl` e `foldr` que só estão definidas para listas com pelo menos um elemento (i.e. não-vazias). `Foldl1` e *foldr1* têm apenas dois argumentos (uma operação de agregação e uma lista) e o seu resultado é dado pelas equações seguintes.

$$\texttt{foldl1}\ (\oplus)\ [x_1,\ldots,x_n] = (\ldots(x_1 \oplus x_2)\ldots) \oplus x_n$$
$$\texttt{foldr1}\ (\oplus)\ [x_1,\ldots,x_n] = x_1 \oplus (\ldots(x_{n-1} \oplus x_n)\ldots)$$

   (a) Mostre que pode definir as funções `maximum`, `minimum` :: Ord $a \Rightarrow [a] \to a$ do prelúdio-padrão (que calculam, respectivamente, o maior e o menor elemento duma lista não-vazia) usando `foldl1` e `foldr1`.

   (b) Mostre que pode definir `foldl1` e `foldr1` usando `foldl` e `foldr`. Sugestão: utilize as funções `head`, `tail`, `last` e `init`.

3. Sem consultar a especificação do Haskell, escreva definições não-recursivas das seguintes funções do prelúdio-padrão:

   (a) $(+\!\!+) :: [a] \to [a] \to [a]$, usando `foldr`;

   (b) `concat` :: $[[a]] \to [a]$, usando `foldr`;

   (c) `reverse` :: $[a] \to [a]$, usando `foldr`;

   (d) `reverse` :: $[a] \to [a]$, usando `foldl`.

4. A função de ordem superior `until` :: $(a \to Bool) \to (a \to a) \to a \to a$ está definida no prelúdio-padrão; `until p f` é a função que repete sucessivamente a aplicação de `f` ao argumento até que que `p` seja verdade. Usando `until`, escreva uma definição não-recursiva da função

$$\texttt{mdc}\ a\ b = \text{if } b == 0 \text{ then } a \text{ else } \texttt{mdc}\ b\ (a\texttt{`mod`}b)$$

   que calcula o máximo divisor comum pelo algoritmo de Euclides.

5. Considere a função `rotate`, que produz todas as rotações possíveis de uma lista. Ou seja `rotate [1, 2, 3] = [[1, 2, 3],[2, 3, 1],[3, 1, 2]]`.

   (a) Defina a função `shift`, que coloca o primeiro elemento de uma lista no final. Ou seja `shift [1,2, 3] = [2, 3, 1]` e `shift "eat" = "ate"`.

   (b) Utilizando `foldr` e `shift` defina a função `rotate`.