Princípios de Programação

Exercícios

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática
Licenciatura em Engenharia Informática

2019/2020

Teste de funções com QuickCheck

1. Construa testes QuickCheck para testar as seguintes propriedades aritméticas:

   (a) Após cada número par vem um número ímpar.

   (b) Os números de Mersenne são dados por $M_n = 2^n - 1$. Na Grécia antiga verificou-se que M_2, M_3, M_5 e M_7 eram números primos. Na altura, conjecturou-se que, para qualquer primo p, M_p era também primo. Defina um teste para tentar refutar esta conjectura. Assuma dada uma função `isPrime :: Integer -> Bool`.

   (c) Defina um teste para tentar refutar a conjectura de Collatz.

   (d) Sabe-se que, para dois números naturais a e b, $\text{mdc}(a,b) \times \text{mmc}(a,b) = a \times b$. Implemente estas duas funções, e valide a sua correção com um teste que verifica esta igualdade.

2. Dada a função `reverse :: [a] -> [a]`, escreva testes que verifiquem se

   (a) o comprimento da lista de entrada e de saída coincidem,

   (b) a inversa da inversa é a lista original,

   (c) a lista inversa é uma permutação da lista original,

   (d) o i-ésimo elemento da lista inversa é igual ao $(n-1-i)$-ésimo elemento da lista original, onde n é o comprimento da lista original.

3. Teste as várias funções descritas no exercício **1** da secção da Recursão (Capítulo 4).

4. Considere o tipo de dados formas geométricas, exercício **1** do Capítulo 7.
   (a) Torne uma instância da classe `Arbitrary`.
   (b) Escreva uma propriedade que assegure que o perímetro é não negativo.
   (c) Considere agora o caso em que a probabilidade de gerar um círculo é o dobro da probabilidade de gerar um triângulo, que por sua vez é o dobro da probabilidade de gerar um rectângulo.

5. Escreva testes que verifiquem o módulo conjuntos ordenados, exercício **1** do capítulo 6.
   (a) Torne o tipo de dados `Set` instância da classe de tipos `Arbitrary`.

   ```
   data Set a = S [a]
   ```

   (b) Classifique as várias operações em *construtoras*, *observadoras* e *derivadas*. Por exemplo, `singleton x` é uma operação derivada porque pode ser definida por `insert x empty`.
   (c) Construa um teste para cada par observadora-construtora.
   (d) Construa um teste para cada operação derivada.
   (e) Construa testes para testar um par construtora-construtora. O que acontece se inserirmos dois valores em sucessão?

6. Considere o seguinte tipo recursivo definido no exercício **2** do capítulo 6.

   ```
   data Tree = Leaf Int | Node Tree Tree deriving Show
   ```

   (a) Torne o tipo `Tree` instancia da classe `Arbitrary`, fazendo uso dos construtores.
   (b) Mesmo exercício utilizando as funções **`liftM`** e **`liftM2`** para gerar os construtores. Utilize a função `oneof` (ou `frequency`) para escolher entre os construtores.
   (c) Mesmo exercício utilizando a função que cria uma árvore a partir de uma lista.
   (d) Se as árvores produzidas forem demasiado grandes (ou se o gerador não terminar), utilize o operador `sized :: (Int -> Gen a)-> Gen a`, disponível no `Quickcheck`, que dado um inteiro limita a dimensão da amostra aleatória.
   (e) Defina um teste para a seguinte propriedade: a lista que resulta de achatar a árvore tem o mesmo tamanho que a árvore original.
   (f) Teste para: inverter uma árvore não altera o seu tamanho.
   (g) Teste para: criar uma árvore a partir de uma lista e achatá-la de novo, mantém os elementos da lista original.