
Linguagens e Ambientes de Programação (2018/2019)

Prática 3

Árvores e outros tipos soma. Exercícios de 21 a 26.

- 21 - Considere o tipo árvore binária em OCaml:

```
type 'a tree = Nil | Node of 'a * 'a tree * 'a tree
```

Escreva as seguintes três funções sobre árvores:

```
howMany : 'a -> 'a tree -> int    (* Conta número de ocorrências do valor na árvore *)
eqPairs : ('a * 'a) tree -> int    (* Conta número de pares com as duas componentes iguais *)
treeToList : 'a tree -> 'a list    (* Converte árvore em lista, por uma ordem qualquer *)
```

- 22 - Programe uma função:

```
balanced: 'a tree -> bool
```

que determine se uma árvore binária está ou não equilibrada. Uma árvore binária diz-se *equilibrada* se, para cada nó, a diferença de profundidades das suas subárvores não superar a unidade. (Copie a função a função auxiliar *height* da teórica 4.)

Exemplos:

```
balanced Nil = true
balanced (Node(3,Node(5,Nil,Nil),Node(6,Nil,Nil))) = true
balanced (Node(1,Nil,Node(2,Nil,Node(3,Nil,Nil)))) = false
```

- 23 - Programe uma função:

```
subtrees: 'a tree -> 'a tree list
```

que produza a lista de todas as subárvores distintas que ocorrem na árvore argumento.

Exemplos:

```
subtrees (Node(5,Nil,Node(6,Nil,Nil))) =
  [Node(5,Nil,Node(6,Nil,Nil)); Node(6,Nil,Nil); Nil]
subtrees Nil = [Nil]
```

- 24 - Primavera. Programe uma função:

```
spring: 'a -> 'a tree -> 'a tree
```

que faça crescer novas folhas em todos os pontos duma árvore onde esteja Nil. As novas folhas são todas iguais entre si.

- 25 - Outono. Programe uma função:

```
fall: 'a tree -> 'a tree
```

que elimine todas as folhas existentes duma árvore. Os nós interiores que ficam a descoberto tornam-se folhas, claro, as quais já não são para eliminar.

- 26 - Repita os exercícios 23, 24 e 25, mas agora para árvores n-árias. O tipo árvore n-ária em OCaml define-se assim:

```
type 'a ntree = NNil | NNode of 'a * 'a ntree list
treeToList : 'a ntree -> 'a list
subtrees : 'a ntree -> 'a ntree list
spring: 'a -> 'a ntree -> 'a ntree
fall: 'a ntree -> 'a ntree
```

Nota: Uma folha duma árvore binária tem dois Nil por baixo, mas uma folha duma árvore n-ária costuma ter apenas uma lista vazia de filhos, portanto sem qualquer Nil.

- 27 - Defina um tipo soma **exp** para representar **expressões algébricas com uma variável real "x"**. Nas expressões devem poder ocorrer os operadores binários "+", "-", "*", "/", o operador unário "-", a variável "x" e ainda literais reais. Exemplos de expressões:

$$2*x^2+5$$

$$2*x^7+5*(x-5)^2-3$$

Depois de definido o tipo, escreva as seguintes funções sobre expressões algébricas:

- a) eval: float -> exp -> float (* Avalia a expressão para um dado valor de "x" *)
 b) deriv: exp -> exp (* Determina a derivada em ordem a "x". Não simplifique o resultado. *)

Ajuda: O tipo exp pode ser definido assim

```
type exp =
  Add of exp*exp (* soma *)
  | Sub of exp*exp (* subtração *)
  | Mult of exp*exp (* multiplicação *)
  | Div of exp*exp (* divisão *)
  | Power of exp*int (* potência *)
  | Sym of exp (* simétrico *)
  | Const of float (* constante*)
  | Var (* variável *)
;;
```

Usando esta representação, a expressão $2*x^2+5$ escreve-se: `Add(Mult(Const 2.0,Power(Var,2)),Const 5.0)`.