Verificador de tautologias

Nos exercícios seguintes pretende-se que se baseie no verificador de tautologias apresentado na aula teórica 14.

7.1 Escreva uma definição da função `satisfaz :: Prop → Bool` que verifica se uma proposição é *satisfazível*, isto é, se existe uma atribuição de valores às variáveis que a torna verdadeira.

7.2 Escreva uma definição da função `equiv :: Prop → Prop → Bool` que verifica se duas proposições são *equivalentes*, isto é, tomam o mesmo valor de verdade para todas as atribuições de variáveis. Sugestão: p, q são equivalentes se e só se $p \implies q \wedge q \implies p$ for uma tautologia.

7.3 Modifique o verificador de tautologias acrescentando uma nova conectiva para equivalência entre proposições.

7.4 Escreva uma definição duma função `showProp :: Prop → String` para converter uma proposição em texto; alguns exemplos:

```
> showProp (Neg (Var 'a'))
"(~a)"
> showProp (Disj (Var 'a') (Conj (Var 'a') (Var 'b')))
"(a || (a && b))"
> showProp (Impl (Var 'a') (Impl (Neg (Var 'a')) (Const False))
"(a -> ((~a) -> F))"
```

7.5 Modifique o verificador de tautologias para imprimir a tabela de verdade duma proposição. Mais precisamente, escreva uma função `tabela :: Prop →` *IO* () que imprime a tabela de verdade da proposição dada.

Árvores de pesquisa

Nos exercícios seguintes considere a definição do tipo de árvores de pesquisa apresentada na aulas teórica 15:

$$\texttt{data Arv } a = \texttt{Vazia} \mid \texttt{No } a \ (\texttt{Arv } a) \ (\texttt{Arv } a)$$

7.6 Escreva uma definição recursiva da função `sumArv :: Num` $a \Rightarrow$ `Arv` $a \to a$ que soma todos os valores duma árvore binária de números.

7.7 Baseado-se na função `listar :: Arv` $a \to [a]$ apresentada na aula teórica, escreva a definição duma função para listar os elementos duma árvore de pesquisa por *ordem decrescente*.

7.8 Escreva uma definição da função `nivel :: Int → Arv` $a \to [a]$ tal que `nivel` n *arv* é a lista ordenada dos valores da árvore no nível n, isto é, a uma altura n (considerando que a raiz tem altura 0).

7.9 Experimente usar o interpretador de Haskell para calcular a altura de árvores de pesquisa com n valores.

(a) usando o método de partições binárias, e.g. `construir` $[1..n]$;

(b) usando inserções simples, e.g. `foldr inserir Vazia` $[1..n]$;

(c) usando inserções AVL, e.g. `foldr inserirAVL Vazia` $[1..n]$;

Experimente com $n = 10$, 100 e 1000 e compare a altura obtida com o minorante teórico: uma árvore binária com n nós tem altura $\geq \log_2 n$.

7.10 Escreva uma definição da função de ordem superior `mapArv` $:: (a \to b) \to$ `Arv` $a \to$ `Arv` b tal que `mapArv` $f\ t$ aplica uma função f a cada valor duma árvore t.

7.11 Neste exercício pretende-se implementar uma variante da remoção de um valor duma árvore de pesquisa simples.

(a) Baseando-se na função `maisEsq` $::$ `Arv` $a \to a$ apresentada na aula teórica, escreva uma definição da função `maisDir` $::$ `Arv` $a \to a$ que obtém o valor mais à direita numa árvore (i.e., o maior valor).

(b) Usando a função da alínea anterior, escreva uma definição alternativa da função `remover` $::$ `Ord` $a \Rightarrow a \to$ `Arv` $a \to$ `Arv` a que use o valor mais à direita da sub-árvore esquerda no caso de um nó com dois descentes não-vazios.

7.12 Escreva uma definição da função `removerAVL` $::$ `Ord` $a \Rightarrow a \to$ `Arv` $a \to$ `Arv` a para remover um valor duma árvore AVL de forma a manter a árvore resultante equilibrada. Sugestão: modifique a função análoga para árvores de pesquisa simples.

7.13 A informação de altura de cada sub-árvore é usada para re-equilibrar as árvores AVL. Neste exercício pretende-se que modifique a implementação apresentada na aula teórica de forma a guardar a esta informação nos nós em vez de a re-calcular todas as vezes que é usada. Começe por alterar a declaração de tipo de árvore de forma a que cada nó tenha um argumento extra para a altura:

$$\texttt{data Arv}\ a\ =\ \texttt{Vazia}\ \mid\ \texttt{No Int}\ a\ (\texttt{Arv}\ a)\ (\texttt{Arv}\ a)$$

Tenha o cuidado de modificar as funções que efectuam rotações de forma a actualizar correctamente a altura.