Cálculo de Programas

Licenciatura em Engenharia Informática

Ficha 7

1. Considere o seguinte tipo de dados:

   $$\mathbf{data}\ \mathsf{LTree}\ a = \mathsf{Leaf}\ a \mid \mathsf{Branch}\ (\mathsf{LTree}\ a)\ (\mathsf{LTree}\ a)$$

   (a) Enuncie e demonstre a lei de fusão dos catamorfismos para este tipo.

   (b) Considere as seguintes funções:

   $$\begin{array}{l} \mathsf{soma} :: \mathsf{LTree\ Int} \to \mathsf{Int} \\ \mathsf{soma}\ (\mathsf{Leaf}\ x) \quad\quad = x \\ \mathsf{soma}\ (\mathsf{Branch}\ l\ r) = \mathsf{soma}\ l + \mathsf{soma}\ r \\ \mathsf{join} :: \mathsf{LTree}\ (\mathsf{LTree}\ a) \to \mathsf{LTree}\ a \\ \mathsf{join}\ (\mathsf{Leaf}\ x) \quad\quad = x \\ \mathsf{join}\ (\mathsf{Branch}\ l\ r) \quad = \mathsf{Branch}\ (\mathsf{join}\ l)\ (\mathsf{join}\ r) \end{array}$$

   Implemente estas funções no estilo *point-free* usando catamorfismos. Desenhe os respectivos diagramas.

   (c) Demonstre que $\mathsf{map}_{\mathsf{LT}}\ f = (\!|\mathsf{in}_{\mathsf{LT}} \circ (f + \mathsf{id})|\!)_{\mathsf{LT}}$ corresponde à seguinte definição *pointwise*:

   $$\begin{array}{l} \mathsf{map}_{\mathsf{LT}} :: (a \to b) \to \mathsf{LTree}\ a \to \mathsf{LTree}\ b \\ \mathsf{map}_{\mathsf{LT}}\ f\ (\mathsf{Leaf}\ x) \quad\quad = \mathsf{Leaf}\ (f\ x) \\ \mathsf{map}_{\mathsf{LT}}\ f\ (\mathsf{Branch}\ l\ r) = \mathsf{Branch}\ (\mathsf{map}_{\mathsf{LT}}\ f\ l)\ (\mathsf{map}_{\mathsf{LT}}\ f\ r) \end{array}$$

   (d) Demonstre que $\mathsf{soma} \circ \mathsf{join} = \mathsf{soma} \circ \mathsf{map}_{\mathsf{LT}}\ \mathsf{soma}$, assumindo que $\mathsf{soma} \circ \mathsf{uncurry\ Branch} = \mathsf{plus} \circ (\mathsf{soma} \times \mathsf{soma})$.

2. Considere o seguinte tipo de dados para representar árvores binárias sem conteúdo.

   $$\mathbf{data}\ \mathsf{STree} = \mathsf{Tip} \mid \mathsf{Fork\ STree\ STree}$$

   (a) Defina as funções $\mathsf{out}_{\mathsf{T}}$ em Haskell no estilo *pointwise* e in_{T} no estilo *point-free*.

   (b) Desenhe o diagrama dos catamorfismos para este tipo, e identifique a respectiva lei universal.

   (c) Defina no estilo *point-free* usando catamorfismos as funções $\mathsf{nfolhas} :: \mathsf{STree} \to \mathsf{Int}$ (conta o número de folhas) e $\mathsf{mirror} :: \mathsf{STree} \to \mathsf{STree}$ (espelha uma árvore).

   (d) Demonstre que $\mathsf{nfolhas} \circ \mathsf{mirror} = \mathsf{nfolhas}$, assumindo que a adição é comutativa, ou seja, $\mathsf{plus} \circ \mathsf{swap} = \mathsf{plus}$.

3. Considere o seguinte tipo de dados para representar listas onde não só é possível inserir elementos à cabeça (Cons), mas também na cauda da lista (Snoc).

   $$\mathbf{data}\ \mathsf{Seq}\ a = \mathsf{Nil} \mid \mathsf{Cons}\ a\ (\mathsf{Seq}\ a) \mid \mathsf{Snoc}\ a\ (\mathsf{Seq}\ a)$$

(a) Atendendo ao isomorfismo $\mathsf{Seq}\ a \cong 1 + (a \times \mathsf{Seq}\ a + a \times \mathsf{Seq}\ a)$, defina as funções out_S em Haskell no estilo *pointwise* e in_S no estilo *point-free*.

(b) Desenhe o diagrama dos catamorfismos para este tipo, e identifique a respectiva lei universal.

(c) A função $\mathsf{folds} :: b \to (a \to b \to b) \to (a \to b \to b) \to \mathsf{Seq}\ a \to b$ pode ser definida como $\mathsf{folds}\ z\ f\ g = (\!|\underline{z} \triangledown (\mathsf{ap} \circ (f \times \mathsf{id}) \triangledown \mathsf{ap} \circ (g \times \mathsf{id}))|\!)_\mathsf{S}$. Derive a respectiva definição no estilo *pointwise*.

(d) Uma das vantagens destas listas é que é possível definir a seguinte função de inversão, que executa em tempo linear.

$$\begin{array}{l}\mathsf{rev} :: \mathsf{Seq}\ a \to \mathsf{Seq}\ a \\ \mathsf{rev}\ \mathsf{Nil} = \mathsf{Nil} \\ \mathsf{rev}\ (\mathsf{Cons}\ x\ l) = \mathsf{Snoc}\ x\ (\mathsf{rev}\ l) \\ \mathsf{rev}\ (\mathsf{Snoc}\ x\ l) = \mathsf{Cons}\ x\ (\mathsf{rev}\ l)\end{array}$$

Defina rev no estilo *point-free* usando um catamorfismo e demonstre que a essa definição é equivalente à apresentada acima.

(e) Demonstre que $\mathsf{rev} \circ \mathsf{rev} = \mathsf{id}$.

4. Considere o seguinte tipo de dados para representar números naturais.

$$\mathbf{data}\ \mathsf{Nat} = \mathsf{Zero} \mid \mathsf{Succ}\ \mathsf{Nat}$$

(a) Defina a função $\mathsf{double} :: \mathsf{Nat} \to \mathsf{Nat}$ no estilo *point-free* usando um catamorfismo.

(b) Verifique, derivando a respectiva versão *pointwise*, que a função $\mathsf{add} :: \mathsf{Nat} \to \mathsf{Nat} \to \mathsf{Nat}$ pode ser definida como $\mathsf{add} = (\!|\underline{\mathsf{id}} \triangledown \overline{\mathsf{Succ} \circ \mathsf{ap}}|\!)_\mathsf{N}$.

(c) Na definição anterior, a recursividade é feita sobre o primeiro argumento. Apresente uma definição alternativa para a função add, usando também um catamorfismo, assumindo que o primeiro parâmetro é fixo e que a recursividade é feita sobre o segundo, ou seja, $\mathsf{add}\ n = (\!|\ldots|\!)_\mathsf{N}$.

5. Considere o seguinte tipo de dados para representar (algumas) expressões *point-free* com variáveis.

$$\mathbf{data}\ \mathsf{Exp} = \mathsf{Var}\ \mathsf{String} \mid \mathsf{Split}\ \mathsf{Exp}\ \mathsf{Exp} \mid \mathsf{Comp}\ [\mathsf{Exp}]$$

(a) Defina as funções out_E em Haskell no estilo *pointwise* e in_E no estilo *point-free*.

(b) Desenhe o diagrama dos catamorfismos para este tipo, e identifique a respectiva lei universal.

(c) Defina, usando um catamorfismo, a funçao $\mathsf{vars} :: \mathsf{Exp} \to [\mathsf{String}]$, que extrai as variáveis de uma expressão.

(d) Defina, usando um catamorfismo, a função $\mathsf{subst} :: (\mathsf{String} \to \mathsf{Exp}) \to \mathsf{Exp} \to \mathsf{Exp}$, que aplica uma substituição a todas as variáveis de uma expressão. Assuma que o primeiro parâmetro é fixo e que a recursividade é feita sobre o segundo, ou seja $\mathsf{subst}\ f = (\!|\ldots|\!)_\mathsf{E}$.

(e) Demonstre que $\mathsf{subst}\ \mathsf{Var} = \mathsf{id}$.