Cálculo de Programas
Exame

Licenciatura em Engenharia Informática

18 Julho de 2009, 9h30

O exame tem a duração de 2h. Cada alínea vale 2 pontos.

1. Identifique o isomorfismo que a seguinte função testemunha:

   $$\mathsf{iso} = (\mathsf{fst} \triangledown \mathsf{fst}) \triangle (\mathsf{snd} + \mathsf{snd})$$

   Desenhe o diagrama respectivo.

2. Demonstre que $\mathsf{iso} \circ \mathsf{inl} = \mathsf{id} \times \mathsf{inl}$.

3. Derive uma definição *pointwise* da função iso a partir da sua definição *point-free*.

4. Defina no estilo *point-free* a seguinte função (sem usar o uncurry):

   $$\mathsf{comp} :: (b \to c, a \to b) \to (a \to c)$$
   $$\mathsf{comp}\ (f, g) = f \circ g$$

   Demonstre que a definição que obteve é equivalente à apresentada.

5. Relembre a definição do tipo From a:

   $$\textbf{data}\ \mathsf{From}\ a = \mathsf{First}\ a \mid \mathsf{Next}\ (\mathsf{From}\ a)$$

   Enuncie a propriedade universal dos catamorfismos para este tipo e demonstre que $\mathsf{out_F} = (\!|\mathsf{id} + \mathsf{in_F}|\!)_\mathsf{F}$.

6. O tipo From a pode ser visto como um *monad*, onde o número de Nexts conta o número de *binds* realizados até ao momento. No estilo *point-free* poderia ser implementado da seguinte forma:

   $$\mathsf{map_F} :: (a \to b) \to (\mathsf{From}\ a \to \mathsf{From}\ b)$$
   $$\mathsf{map_F}\ f = (\!|\mathsf{in_F} \circ (f + \mathsf{id})|\!)_\mathsf{F}$$
   $$\mathsf{return} :: a \to \mathsf{From}\ a$$
   $$\mathsf{return} = \mathsf{First}$$
   $$\mathsf{join} :: \mathsf{From}\ (\mathsf{From}\ a) \to \mathsf{From}\ a$$
   $$\mathsf{join} = (\!|\mathsf{id} \triangledown \mathsf{Next}|\!)_\mathsf{F}$$

   Derive uma implementação *pointwise* explicitamente recursiva e **eficiente** do operador *bind* a partir da sua definição *point-free*:

   $$(\gg\!= f) = \mathsf{join} \circ \mathsf{map_F}\ f$$

7. É possível definir uma função para extrair o conteúdo e o número de *binds* do *monad* From a da seguinte forma:

   $$\mathsf{run} :: \mathsf{From}\ a \to a \times \mathsf{Int}$$
   $$\mathsf{run} = (\!|\mathsf{id} \triangledown \mathsf{id}|\!)_\mathsf{F} \triangle (\!|\underline{0} \triangledown \mathsf{succ}|\!)_\mathsf{F}$$

   Defina a função inversa $\mathsf{run}^{-1} :: a \times \mathsf{Int} \to \mathsf{From}\ a$ usando um anamorfismo para o tipo From a. Pode implementar o gene no estilo *pointwise*. Não se esqueça de desenhar o respectivo diagrama.

8. Considere o seguinte tipo para representar expressões com operadores de aridade variável:

   data Exp a = Var a | Op String [Exp a]

   O parâmetro a determina o tipo das variáveis. Determine o tipo isomorfo a Exp a e codifique as respectivas funções $\mathsf{in_E}$ e $\mathsf{out_E}$ nos estilos *point-free* e *pointwise*, respectivamente.

9. Este tipo também pode ser visto como um *monad*, servindo a operação de *bind* para fazer a substituição:

   $(\gg\!=)$ :: Exp $a \to (a \to$ Exp $b) \to$ Exp b
   Var $x \gg\!= f = f\ x$
   Op $o\ l \gg\!= f$ = Op o (map $(\gg\!= f)\ l$)

   Defina $(\gg\!= f)$ :: Exp $a \to$ Exp b como um catamorfismo no estilo *point-free*. Não se esqueça de desenhar o respectivo diagrama.

10. Demonstre que a seguinte propriedade é válida para qualquer monad:

    return $x \gg\!= f = f\ x$

Boa sorte!