Cálculo de Programas
Teste

Licenciatura em Engenharia Informática

29 Junho de 2009, 9h00

O teste tem a duração de 2h. As 5 primeiras questões valem 2 pontos cada; as restantes 2.5 pontos cada.

1. Identifique o isomorfismo que a seguinte função testemunha:

   $$\mathsf{iso} = (! + !) \triangle (\mathsf{id} \triangledown \mathsf{id})$$

   Desenhe o diagrama respectivo.

2. Demonstre que $\mathsf{iso} \circ \mathsf{inl} = (\mathsf{inl} \circ !) \triangle \mathsf{id}$.

3. Defina no estilo *point-free* a função iso^{-1}. Desenhe o diagrama respectivo.

4. Considere o seguinte tipo:

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

   Determine o tipo polinomial isomorfo a $\mathsf{From}\ a$ e codifique as respectivas funções $\mathsf{in_F}$ e $\mathsf{out_F}$ nos estilos *point-free* e *pointwise*, respectivamente.

5. Enuncie e demonstre a lei de fusão dos catamorfismos para o tipo $\mathsf{From}\ a$.

6. Considere a seguinte implementação eficiente da função *reverse*:

   ```
   rev :: [a] → [a]
   rev l = aux (l, [ ])
     where aux :: ([a], [a]) → [a]
       aux ([ ], l)    = l
       aux (h : t, l) = aux (t, h : l)
   ```

   Codifique a função aux usando um hilomorfismo com tipo intermédio $\mathsf{From}\ [a]$. Pode codificar os genes no estilo *pointwise*. Não se esqueça de desenhar o diagrama respectivo.

7. O tipo $\mathsf{From}\ a$ 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:

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

   Demonstre que este tipo satisfaz a seguinte propriedade dos *monads*:

   $$\mathsf{join} \circ \mathsf{join} = \mathsf{join} \circ \mathsf{map_F}\ \mathsf{join}$$

   Assuma que $\mathsf{join} \circ \mathsf{Next} = \mathsf{Next} \circ \mathsf{join}$.

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

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

   Derive uma versão *pointwise* explicitamente recursiva e **eficiente** da função run.

9. Dada uma sequência de bytes é possível definir uma cifra sequencial muito simples, onde a cifragem de um byte é feita à custa de um ou exclusivo com o byte anterior. O valor que é usado para cifrar o primeiro byte serve de chave. O comportamento pretendido é o seguinte:

$$\begin{aligned}&\mathsf{cifra} :: \mathsf{Byte} \to [\mathsf{Byte}] \to [\mathsf{Byte}]\\&\mathsf{cifra}\ c\ [b_0, b_1, \ldots] = [b_0\ `\mathsf{xor}`\ c, b_1\ `\mathsf{xor}`\ b_0, b_2\ `\mathsf{xor}`\ b_1, \ldots]\end{aligned}$$

   Esta função pode ser codificada de forma muito simples usando o *monad* estado, onde o estado é usado para armazenar o byte anterior enquanto se faz uma travessia da sequência de bytes da mensagem:

$$\begin{aligned}&\mathsf{cifra} :: \mathsf{Byte} \to [\mathsf{Byte}] \to [\mathsf{Byte}]\\&\mathsf{cifra}\ c\ l = \mathsf{evalState}\ (\mathsf{sequence}\ (\mathsf{map}\ \mathsf{aux}\ l))\ c\end{aligned}$$

   Defina a função auxiliar aux por forma a obter o comportamento especificado acima. Não se esqueça de definir o seu tipo.

Boa sorte!