Cálculo de Programas

2.º ano da LEI (Universidade do Minho)
Ano Lectivo 2010/2011

Método A, Turno TP 1 — Ficha 1
(Sem consulta)

Questão 1 1. Identifique o isomorfismo testemunhado pela função:

$$[< id, i_2 \,.\, ! > \,,\; id \times i_1]$$

Construa o respectivo diagrama.

2. Recorra à lei da troca para re-escrever a função anterior sob a forma de um *split*.

Questão 2 Demonstre a seguinte propriedade do combinador condicional de McCarthy:

$$p \rightarrow f.g \;,\; h.k \;=\; [f,h] \,.\, (\, p \rightarrow i1.g \;,\; i2.k \,)$$

Número: ____________ **Nome:** ____

Cálculo de Programas

2.º ano da LEI (Universidade do Minho)
Ano Lectivo 2010/2011

Método A, Turno TP 3 — Ficha 1
(Sem consulta)

Questão 1

Relembre as seguintes definições:

$$swap = < \pi_2,\ \pi_1 > \quad \text{e} \quad coswap = [\ i_2,\ i_1\ ].$$

1. Identifique o isomorfismo testemunhado pela função: $coswap \,.\, (swap + swap)$ desenhando o respectivo diagrama.
2. Formule a lei natural da função anterior, com recurso ao diagrama respectivo e demonstre-a analíticamente. (Sugestão: use as leis naturais do *coswap* e do *swap*)

Questão 2 Demonstre a seguinte propriedade do combinador condicional de McCarthy:

$$(\neg \,.\, p) \rightarrow g\ ,\ f \;=\; (\ p \rightarrow f\ ,\ g\ )$$

sabendo que é válida a propriedade:

$$(\neg \,.\, p\ )? \;=\; [\ i_2,\ i_1\ ] \,.\, (p?)$$

Número: ____________ **Nome:** ____________________________________

Cálculo de Programas

1.º Ano da Licenciatura em Engenharia Informática (Universidade do Minho)
Ano Lectivo de 20010/11

Avaliação Individual (Método A) —- Ficha nr. 1

IDENTIFICAÇÃO DO ALUNO:

Nome: .... **Número** : | | | | | |

PROVA SEM CONSULTA (30 minutos)

Questão 1 Considere a definição

$$t \;=\; \langle [i_2, i_1] \cdot \pi_2, [i_2, i_1] \cdot \pi_1 \rangle$$

1. Calcule o tipo mais geral de t.
2. Enuncie através de um diagrama a propriedade *natural* de t e demonstre-a.

Questão 2 Demonstre a seguinte propriedade do combinador condicional de McCarthy

$$(\neg \cdot p) \rightarrow f, g \quad = \quad p \rightarrow g, f$$

sabendo que $(\neg \cdot p)? = [i_2, i_1] \cdot (p)?$

Cálculo de Programas

2.º Ano da Lic. em Engenharia Informática (LEI) da Universidade do Minho
Ano Lectivo de 2010/11

Avaliação Individual Nr. 1 (Turno TP-2, Método A)

PROVA SEM CONSULTA (30 minutos)

Questão 1 Sendo iso a única função que é solução do diagrama ao lado, enuncie, também com base num diagrama, a propriedade natural de iso e calcule a seguinte definição sua, em Haskell:

$$
\begin{tikzcd}
A+1 \arrow[r, "i_1"] \arrow[dr, "i_2+!"'] & (A+1)+B \arrow[d, "iso"] & B \arrow[l, "i_2"'] \arrow[dl, "i_1 \cdot i_1"] \\
& (B+A)+1 &
\end{tikzcd}
$$

$$
\begin{aligned}
&iso\ (Left\ (Left\ a)) = Left\ (Right\ a)\\
&iso\ (Left\ (Right\ \_)) = Right\ ()\\
&iso\ (Right\ b) = Left\ (Left\ b)
\end{aligned}
$$

RESOLUÇÃO PROPOSTA: Propriedade natural: seguindo a estratégia estudada nas aulas, parte-se do tipo $iso :: (A+1)+B \to (B+A)+1$ e começa-se por desenhar

$$
\begin{tikzcd}
(B+A)+1 & (A+1)+B \arrow[l, "iso"'] \\
(D+C)+1 & (C+1)+D \arrow[l, "iso"]
\end{tikzcd}
$$

que se fecha com as funções $(g+f)+id$ e $(f+id)+g$ obtidas a partir de $(B+A)+1$ e $(A+1)+B$ substituindo $A, B, 1 := f, g, id$. Logo:

$$
\begin{tikzcd}
(B+A)+1 \arrow[d, "(g+f)+id"'] & (A+1)+B \arrow[l, "iso"'] \arrow[d, "(f+id)+g"] \\
(D+C)+1 & (C+1)+D \arrow[l, "iso"]
\end{tikzcd}
\qquad \text{isto é} \qquad ((g+f)+id)\cdot iso = iso\cdot((f+id)+g)
$$

Cálculo de iso em Haskell: iso é tal que, como o diagrama dado indica, $iso \cdot i_1 = i_2+!$ e $iso \cdot i_2 = i_1 \cdot i_1$. Logo:

$$
\begin{aligned}
&\quad \left\{\begin{array}{l} iso \cdot i_1 = i_2+! \\ iso \cdot i_2 = i_1 \cdot i_1 \end{array}\right.\\
\equiv &\quad \{\ \text{Def-+ (21)}\ \}\\
&\quad \left\{\begin{array}{l} iso \cdot i_1 = [i_1 \cdot i_2\ , i_2 .!] \\ iso \cdot i_2 = i_1 \cdot i_1 \end{array}\right.\\
\equiv &\quad \{\ \text{Universal-+ (16)}\ \}\\
&\quad \left\{\begin{array}{l} iso \cdot i_1 \cdot i_1 = i_1 \cdot i_2 \\ iso \cdot i_1 \cdot i_2 = i_2 .! \\ iso \cdot i_2 = i_1 \cdot i_1 \end{array}\right.\\
\equiv &\quad \{\ \text{Igualdade extensional (4) 3x}\ \}\\
&\quad \left\{\begin{array}{l} (iso \cdot i_1 \cdot i_1)\ a = (i_1 \cdot i_2)\ a \\ (iso \cdot i_1 \cdot i_2)\ x = (i_2 .!)\ x \\ (iso \cdot i_2)\ b = (i_1 \cdot i_1)\ b \end{array}\right.\\
\equiv &\quad \{\ \text{Assoc-comp (2) 2x ; Def-comp (70) 8x}\ \}
\end{aligned}
$$

$$
\begin{array}{ll}
 & iso\ (i_1\ (i_1\ a)) = i_1\ (i_2\ a) \\
 & iso\ (i_1\ (i_2\ x) = i_2\ (!x) \\
 & iso\ (i_2\ b) = i_1\ (i_1\ b) \\
\equiv & \quad \{\ \mathit{Left} = i_1 \text{ e } \mathit{Right} = i_2\ ;\ !x = ()\ \} \\
 & iso\ (\mathit{Left}\ (\mathit{Left}\ a)) = \mathit{Left}\ (\mathit{Right}\ a) \\
 & iso\ (\mathit{Left}\ (\mathit{Right}\ x) = \mathit{Right}\ () \\
 & iso\ (\mathit{Right}\ b) = \mathit{Left}\ (\mathit{Left}\ b)
\end{array}
$$

Questão 2 Partindo da lei

$$\langle f, (p \to g, h)\rangle \ = \ p \to \langle f, g\rangle, \langle f, h\rangle \tag{1}$$

demonstrada nas aulas práticas e de outras propriedades do condicional de McCarthy que conhece, mostre que

$$f \times (p \to g, h) \ = \ p \cdot \pi_2 \to f \times g, f \times h$$

RESOLUÇÃO PROPOSTA:

$$
\begin{array}{ll}
 & f \times (p \to g, h) \\
= & \quad \{\ \text{Def-}\times\ (10)\ \} \\
 & \langle f \cdot \pi_1, (p \to g, h) \cdot \pi_2\rangle \\
= & \quad \{\ \text{Segunda lei de fusão do condicional (55)}\ \} \\
 & \langle f \cdot \pi_1, p \cdot \pi_2 \to g \cdot \pi_2, h \cdot \pi_2\rangle \\
= & \quad \{\ \text{Lei (1) do enunciado}\ \} \\
 & p \cdot \pi_2 \to \langle f \cdot \pi_1, g \cdot \pi_2\rangle, \langle f \cdot \pi_1, h \cdot \pi_2\rangle \\
= & \quad \{\ \text{Def-}\times\ (10)\ \text{2x}\ \} \\
 & p \cdot \pi_2 \to f \times g, f \times h
\end{array}
$$

Cálculo de Programas

2.º ano da LEI (Universidade do Minho)
Ano Lectivo 2010/2011

Método A, Turno TP 1 — Ficha 2
Duração: 30 minutos (Sem consulta)

Questão 1 Considere o seguinte tipo de dados:

$LTree\ \ a = Leaf\ \ a\ \mid\ Fork\ \ (LTree\ \ a, LTree\ \ a)$
e as funções seguintes definidas como catamorfismos sobre $Leaf\ \ Trees$:

$listify = (\!|\ [singl, \widehat{(+\!+)}]\ |\!)$
$mirror = (\!|\ in.(id + swap)\ |\!)$
onde $singl\ \ x = [x]$

1. Desenhe o diagrama que representa o catamorfismo $listify$.
2. Derive a versão *pointwise* desta função.

Questão 2 No contexto da questão anterior, prove usando a lei fusão-cata, entre outras, que a seguinte propriedade se verifica:

$sum\ .\ mirror = sum$

sabendo que sum calcula a soma de todas as folhas da árvore.

Número: ______________ **Nome:** ____

Cálculo de Programas

2.º ano da LEI (Universidade do Minho)
Ano Lectivo 2010/2011

Método A, Turno TP 3 — Ficha 2
Duração: 30 minutos (Sem consulta)

Questão 1 Considere o seguinte tipo de dados:

data $NEList\ \ a = Sing\ \ a\ \mid\ Add\ \ (a, NEList\ \ a)$

e a função $filter\ p$ sobre $NEList\ \ a$:

```
filter :: (a ->Bool) -> NEList a -> [a]
filter p (Sing x) | (p x) = [x]
                  | otherwise = []
filter p (Add (x,xs)) | (p x) = x:(filter p xs)
                      | otherwise = filter p xs
```

1. A função $filter\ p$ pode ser definida como um catarmorfismo sobre $NEList\ a$:

   $$filter\ p = (\![[(p \rightarrow singl, nil), (p.\pi_1 \rightarrow cons, \pi_2)]]\!)$$

   onde $singl\ x = [x]$, $cons\ (x, xs) = x : xs$ e $nil\ \_ = []$.
   Desenhe o diagrama correspondente.

2. Prove que a versão *pointwise* da função $filterp$ é equivalente ao catamofismo apresentado.

Questão 2 Considere a função $dmap$ ("double map"):

```
dmap :: (a->b)->(a->b)->[a]->([b],[b])
dmap f g = <f1,f2>  where
    f1 [] = []
    f1 (x:xs) = (f x): f2 xs
    f2 [] = []
    f2 (x:xs) = (g x): f1 xs
```

que aplica alternadamente as funções f e g aos elementos de uma lista. Converta `dmap f g` num catamorfismo aplicando-lhe a lei da recursividade múltipla.

Número: __________ **Nome:** ____

Cálculo de Programas

1.º Ano da Licenciatura em Engenharia Informática (Universidade do Minho)
Ano Lectivo de 20010/11

Avaliação Individual (Método A) —- Ficha nr. 2

IDENTIFICAÇÃO DO ALUNO:

Nome: .... **Número** : | | | | | |

PROVA SEM CONSULTA (30 minutos)

Questão 1 Considere a declaração usual do tipo de dados árvores binárias:

```
data BTree a  = Empty | Node (a, (BTree a, BTree a))
```

e a seguinte função:

$$\begin{array}{rcl} (\mathsf{tmap}\, f)\, \mathsf{Empty} & = & \mathsf{Empty} \\ (\mathsf{tmap}\, f)\, \mathsf{Node}(x, l, r) & = & \mathsf{Node}((f\, x), ((\mathsf{tmap}\, f)\, l, (\mathsf{tmap}\, f)\, r) \end{array}$$

1. Escreva a função $\mathsf{tmap}\, f$ como um catamorfismo, começando por desenhar o diagrama correspondente.
2. Mostre que
$$(\mathsf{tmap}\, f) \cdot (\mathsf{tmap}\, g) \;=\; \mathsf{tmap}\, (f.g)$$

Questão 2 Considere a função

$$\mathsf{partition}_p : \quad A^* \longrightarrow A^* \times A^*$$

que divide uma lista em duas, separando os elementos que satisfazem o predicado $p : A \longrightarrow \mathbf{2}$ daqueles que não o fazem. Escreva esta função como um catamorfismo de listas, não se esquecendo de desenhar o diagrama respectivo.

Cálculo de Programas

2.º Ano da Lic. em Engenharia Informática (LEI) da Universidade do Minho
Ano Lectivo de 2010/11

Avaliação Individual Nr. 2 (Turno TP-2, Método A)

PROVA SEM CONSULTA (2 questões — 30 minutos)

Questão 1 É fácil mostrar que o par de funções f e g mutuamente recursivas

$$\left\{\begin{array}{l} f\ 0 = 1 \\ f\ (n+1) = succ\ (f\ n) \end{array}\right. \qquad \left\{\begin{array}{l} g\ 0 = [\,] \\ g\ (n+1) = f\ n : g\ n \end{array}\right.$$

é equivalente ao sistema de equações

$$\left\{\begin{array}{l} f \cdot \mathbf{in} = [\underline{1}\ , succ \cdot \pi_1] \cdot (id + \langle f, g \rangle) \\ g \cdot \mathbf{in} = inlist \cdot (id + \langle f, g \rangle) \end{array}\right.$$

onde $\mathbf{in} = [\underline{0}\ , succ]$, $succ\ n = n + 1$, $inlist = [nil\ , cons]$, $nil = \underline{[\,]}$ e $cons\ (a, b) = a : b$.

Mostre agora, recorrendo às leis da recursividade múltipla e da troca, que f e g se podem combinar num único ciclo-for com duas variáveis

$$loop = \mathsf{for}\ \langle succ \cdot \pi_1, cons \rangle\ (1, [\,])$$

tal que

$$f = \pi_1 \cdot loop$$
$$g = \pi_2 \cdot loop$$

Sugestão: recorde que catamorfismos de naturais são ciclos-for:

$$(\![\underline{i}\ , b]\!) \quad = \quad \mathsf{for}\ b\ i \tag{1}$$

RESOLUÇÃO PROPOSTA: Mostra-se como a definição de $loop$ como um ciclo-for é equivalente à definição mútua de f e g:

$$\begin{array}{cl}
& loop = \mathsf{for}\ \langle succ \cdot \pi_1, cons \rangle\ (1, [\,]) \\
\equiv & \quad \{\ (1) \text{ do enunciado } \} \\
& loop = (\![\underline{(1, [\,])}\ , \langle succ \cdot \pi_1, cons \rangle]\!) \\
\equiv & \quad \{\ \underline{(a, b)} = \langle \underline{a}, \underline{b} \rangle;\ \underline{[\,]} = nil; \text{ lei da troca (27) } \} \\
& loop = (\!\langle [\underline{1}\ , succ \cdot \pi_1], [nil\ , cons] \rangle\!) \\
\equiv & \quad \{\ \text{por Univ-}\times \text{ (5) } loop = \langle f, g \rangle, \text{ pois } \pi_1 \cdot loop = f \text{ e } \pi_2 \cdot loop = g\ \} \\
& \langle f, g \rangle = (\!\langle [\underline{1}\ , succ \cdot \pi_1], [nil\ , cons] \rangle\!) \\
\equiv & \quad \{\ \text{recursividade múltipla (42) para naturais, isto é, para } \mathsf{F}\ f = 1 + f\ \} \\
& \left\{\begin{array}{l} f \cdot \mathbf{in} = [\underline{1}\ , succ \cdot \pi_1] \cdot (id + \langle f, g \rangle) \\ g \cdot \mathbf{in} = [nil\ , cons] \cdot (id + \langle f, g \rangle) \end{array}\right. \\
\equiv & \quad \{\ inlist = [nil\ , cons]\ \} \\
& \left\{\begin{array}{l} f \cdot \mathbf{in} = [\underline{1}\ , succ \cdot \pi_1] \cdot (id + \langle f, g \rangle) \\ g \cdot \mathbf{in} = inlist \cdot (id + \langle f, g \rangle) \end{array}\right.
\end{array}$$

Questão 2 Considere três catamorfismos de listas:

$$count = (\![\underline{0}\ , succ \cdot \pi_2]\!)$$
$$filt\ p = (\![nil\ , p \cdot \pi_1 \rightarrow cons, \pi_2]\!)$$
$$cfilt\ p = (\![\underline{0}\ , p \cdot \pi_1 \rightarrow succ \cdot \pi_2, \pi_2]\!)$$

Complete a demonstração que em baixo se inicia de um facto

$$count \cdot (filt\ p) \quad = \quad cfilt\ p \tag{2}$$

que relaciona esses três catamorfismos:

$$count \cdot (filt\ p) = cfilt\ p$$

$\Leftarrow$ { lei de fusão-cata }

. . . .

$\equiv$ { . . . . }

$$\left\{ \begin{array}{l} count \cdot nil = \underline{0} \\ p \cdot \pi_1 \rightarrow count \cdot cons, count \cdot \pi_2 = (p \cdot \pi_1 \rightarrow succ \cdot \pi_2, \pi_2) \cdot (id \times count) \end{array} \right.$$

$\equiv$ { preencher acima e completar a prova no espaço deixado livre a seguir }

$\vdots$

RESOLUÇÃO PROPOSTA:

$$count \cdot (filt\ p) = cfilt\ p$$

$\Leftarrow$ { lei de fusão-cata (39) }

$$count \cdot [nil\ , p \cdot \pi_1 \rightarrow cons, \pi_2] = [\underline{0}\ , p \cdot \pi_1 \rightarrow succ \cdot \pi_2, \pi_2] \cdot (id + id \times count)$$

$\equiv$ { fusão-+ (19); condicional (54); absorção-+ (20); natural-id (1); Eq-+ (26) }

$$\left\{ \begin{array}{l} count \cdot nil = \underline{0} \\ p \cdot \pi_1 \rightarrow count \cdot cons, count \cdot \pi_2 = (p \cdot \pi_1 \rightarrow succ \cdot \pi_2, \pi_2) \cdot (id \times count) \end{array} \right.$$

$\equiv$ { condicional (55) }

$$\left\{ \begin{array}{l} count \cdot nil = \underline{0} \\ p \cdot \pi_1 \rightarrow count \cdot cons, count \cdot \pi_2 = p \cdot \pi_1 \cdot (id \times count) \rightarrow succ \cdot \pi_2 \cdot (id \times count), \pi_2 \cdot (id \times count) \end{array} \right.$$

$\equiv$ { natural-π_1 (11) e natural-π_2 (12) ; natural-id (1) }

$$\left\{ \begin{array}{l} count \cdot nil = \underline{0} \\ p \cdot \pi_1 \rightarrow count \cdot cons, count \cdot \pi_2 = p \cdot \pi_1 \rightarrow succ \cdot \pi_2 \cdot (id \times count), count \cdot \pi_2 \end{array} \right.$$

$\Leftarrow$ { Leibniz (3) }

$$\left\{ \begin{array}{l} count \cdot nil = \underline{0} \\ count \cdot cons = succ \cdot \pi_2 \cdot (id \times count) \end{array} \right.$$

$\equiv$ { catamorfismo dado }

$$count = (\![\underline{0}\ , succ \cdot \pi_2]\!)$$

Cálculo de Programas

2.º ano da LEI (Universidade do Minho)
Ano Lectivo 2010/2011

Método A, Turno TP 1 — Ficha 3
Duração: 30 minutos (Sem consulta)

Questão 1 Relembre o tipo de dados árvores binárias:

data $BTree\ \ a = Empty \mid Node\ \ a\ \ (LTree\ \ a, LTree\ \ a)$
e a função seguinte definida sobre árvores binárias:

$in_{BT} = [\ Empty, Node\ ]$

1. Considere a seguinte função:
   $g : \mathcal{N}_0 \longrightarrow 1 + \mathcal{N}_0 \times (\mathcal{N}_0 \times \mathcal{N}_0).$
   $g = (\,! + < id, < pred, pred >>).((== 0)?)$
   onde
   $pred : \mathcal{N}_0 \longrightarrow \mathcal{N}_0$
   $pred\, n = n - 1.$

   Descreva o que faz o anamorfismo $[\!(\ g\ )\!]$ sobre árvores binárias, desenhando o respectivo diagrama.
2. Derive a versão *pointwise* desta função.

Questão 2 Prove a lei da absorção da exponenciação através de diagramas e por cálculo analítico.

Número: __________ **Nome:** ________________________________

Cálculo de Programas

2.º ano da LEI (Universidade do Minho)
Ano Lectivo 2010/2011

Método A, Turno TP 3 — Ficha 3
Duração: 30 minutos (Sem consulta)

Questão 1 Considere a seguinte função:

$f = [\!|\ [\ \underline{1}, \widehat{(+)}\ ], h\ |\!]$ where

```
h n  | n<2 = i1 ()
     | otherwise = i2 (pred n, pred (pred n))
pred n = n-1
```

1. Derive a versão *pointfree* da função h.
2. Desenhe o diagrama do hilomorfismo apresentado. O que faz a função f?
3. Derive a versão *pointwise* função f.

Questão 2 Relembre a definição:

$exp :: (a \to b) \to ((c \to a) \to (c \to b))$
$exp\ f = \overline{f.ap}$.

1. Mostre que $(exp\ f).(exp\ g) = exp(f.g)$ através de diagramas.
2. Mostre que $(exp\ f).(exp\ g)\ \ h = f.g.h$

Número: __________ **Nome:** ______________________________

Cálculo de Programas

1.º Ano da Licenciatura em Engenharia Informática (Universidade do Minho)
Ano Lectivo de 20010/11

Avaliação Individual (Método A) —- Ficha nr. 3

IDENTIFICAÇÃO DO ALUNO:

Nome: .... **Número** : | | | | | |

PROVA SEM CONSULTA (30 minutos)

Questão 1 Mostre que

$$\overline{f \cdot g} = \overline{f \cdot \mathsf{ap}} \cdot \overline{g}$$

e indique os tipos das funções f e g.

Questão 2 Considere o seguinte diagrama de um hilomorfismo:

$$\begin{tikzcd}
\mathbb{N} & & \mathbb{N} + (\mathbb{N} \times \mathbb{N}) \arrow[ll, "f"'] \\
\cdots \arrow[u, "\llparenthesis f \rrparenthesis"] & & \cdots \arrow[u, "id + (\llparenthesis f \rrparenthesis \times \llparenthesis f \rrparenthesis)"'] \\
\mathbb{N} \times \mathbb{N} \arrow[u, "{[\!(g)\!]}"] \arrow[r, "g"] & \mathbb{N} + ((\mathbb{N} \times \mathbb{N}) \times (\mathbb{N} \times \mathbb{N})) \arrow[ur, phantom] & \\
\end{tikzcd}$$

onde

$$\begin{array}{rcl}
f & = & [id, \times] \\
g\,(n, m) & = & \mathsf{if}\ n = m\ \mathsf{then}\ \iota_1\, n\ \mathsf{else}\ \iota_2\,((n, k), (k + 1, m)) \\
k\,(n, m) & = & \mathsf{div}_2(n + m)
\end{array}$$

Preencha as reticências no diagrama e diga, justificadamente, qual o tipo e o propósito da seguinte função:

$$\begin{array}{rcl}
h\,0 & = & 1 \\
h\,x & = & \llbracket f, g \rrbracket\,(1, x)
\end{array}$$

Cálculo de Programas

2.º Ano da Lic. em Engenharia Informática (LEI) da Universidade do Minho
Ano Lectivo de 2010/11

Avaliação Individual Nr. 3 (Turno TP-2, Método A)

PROVA SEM CONSULTA (2 questões — 30 minutos)

Questão 1 O combinador

$$
\begin{array}{l}
\mathit{flip} :: (a \to b \to c) \to b \to a \to c \\
\mathit{flip}\ f\ x\ y = f\ y\ x
\end{array}
$$

troca a ordem dos argumentos de uma função. É fácil de ver que flip é um isomorfismo de exponenciais:

$$
\begin{array}{ccccccc}
(C^B)^A & \cong & C^{A\times B} & \cong & C^{B\times A} & \cong & (C^A)^B \\
f & \mapsto & \widehat{f} & \mapsto & \widehat{f}\cdot \mathit{swap} & \mapsto & \overline{\widehat{f}\cdot \mathit{swap}} = \mathit{flip}\ f
\end{array}
$$

Apresente justificações para os passos seguintes do cálculo desse isomorfismo a partir da sua definição ao ponto (*pointwise*):

$$
\begin{array}{ll}
& \mathit{flip}\ f\ x\ y = f\ y\ x \\
\equiv & \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& ap\ (\mathit{flip}\ f\ x, y) = ap\ (f\ y, x) \\
\equiv & \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& (ap \cdot (\mathit{flip}\ f \times id))\ (x, y) = (ap \cdot (f \times id))\ (y, x) \\
\equiv & \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& ap \cdot (\mathit{flip}\ f \times id) = ap \cdot (f \times id) \cdot \mathit{swap} \\
\equiv & \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& \mathit{flip}\ f = \overline{ap \cdot (f \times id) \cdot \mathit{swap}} \\
\equiv & \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& \mathit{flip}\ f = \overline{ap \cdot (\overline{\widehat{f}} \times id) \cdot \mathit{swap}} \\
\equiv & \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& \mathit{flip}\ f = \overline{\widehat{f} \cdot \mathit{swap}}
\end{array}
$$

RESOLUÇÃO PROPOSTA:

1. Definição $ap\ (f, a) = f\ a$ (2×)
2. Def-× (76), composição (70) e natural-id (1), nos dois lados da igualdade
3. $(y, x) = \mathit{swap}\ (x, y)$, seguida de igualdade extensional (4)
4. Universal-exp (28) para $k := \mathit{flip}\ f$
5. curry e $\mathit{uncurry}$ são isomorfismos, logo $\overline{\widehat{f}} = f$
6. Cancelamento-exp (29) para $f := \widehat{f}$.

Questão 2 Apesar das suas semelhanças, os combinadores

$$\begin{array}{lcl}
\mathit{foldr} :: (a \to b \to b) \to b \to [\,a\,] \to b & & \mathit{foldl} :: (b \to a \to b) \to b \to [\,a\,] \to b \\
\mathit{foldr}\ g\ b\ [\,] = b & \text{e} & \mathit{foldl}\ g\ b\ [\,] = b \\
\mathit{foldr}\ g\ b\ (a : s) = g\ a\ (\mathit{foldr}\ g\ b\ s) & & \mathit{foldl}\ g\ b\ (a : s) = \mathit{foldl}\ g\ (g\ b\ a)\ s
\end{array}$$

têm estrutura algorítmica bastante diferente: enquanto $\mathit{foldr}\ \ g\ \ b$ é o catamorfismo de listas $(\!|[\underline{b}\ ,\widehat{g}]|\!)$, $\mathit{foldl}\ \ g$ é o *currying* de um hilomorfismo:

$$\begin{array}{l}
\mathit{foldl} :: (b \to a \to b) \to b \to [\,a\,] \to b \\
\mathit{foldl}\ g = \overline{h} \\
\quad \textbf{where}\ h = \mathit{conquer} \cdot (\mathit{id} + h) \cdot \mathit{divide} \\
\qquad \mathit{divide}\ (b, [\,]) = i_1\ b \\
\qquad \mathit{divide}\ (b, a : s) = i_2\ (g\ b\ a, s) \\
\qquad \mathit{conquer} = \ldots
\end{array}$$

Desenhe o diagrama do hilomorfismo h e infira o gene $\mathit{conquer}$.

RESOLUÇÃO PROPOSTA: Antes de mais, temos que determinar o tipo do prórpio h. Sabendo-se que $\mathit{foldl}\ \ g = \overline{h}$, ter-se-á $h = \widehat{\mathit{foldl}\ g}$. Da definição de foldl infere-se $\mathit{foldl}\ g :: b \to [\,a\,] \to b$, logo $h = \widehat{\mathit{foldl}\ g} :: b \times [\,a\,] \to b$. Daí um primeiro esboço:

$$\begin{tikzcd}
b \times [\,a\,] \arrow[r, "\mathit{divide}"] \arrow[d, "h"'] & \cdots + b \times [\,a\,] \arrow[d, "\mathit{id}+h"] \\
b & \cdots + b \arrow[l, "\mathit{conquer}"]
\end{tikzcd}$$

Falta apenas preencher as reticências e determinar $\mathit{conquer}$. A cláusula $\mathit{divide}\ \ (b, [\,]) = i_1\ \ b$ força claramente um tipo da forma $\mathit{divide} :: b + \ldots$, logo:

$$\begin{tikzcd}
b \times [\,a\,] \arrow[r, "\mathit{divide}"] \arrow[d, "h"'] & b + b \times [\,a\,] \arrow[d, "\mathit{id}+h"] \\
b & b + b \arrow[l, "\mathit{conquer}"]
\end{tikzcd}$$

Como nada sabemos sobre o tipo b, ter-se-á $\mathit{conquer} = [\mathit{id}\ ,\mathit{id}]$. Isso é confirmado exprimindo $\mathit{foldl}\ g$ em termos do próprio h, assumindo g no contexto e explicitando essas identidades:

$$\begin{array}{l}
h :: (b, [\,a\,]) \to b \\
h\ (b, [\,]) = \mathit{id}\ b \\
h\ (b, a : s) = \mathit{id}\ (h\ (g\ b\ a, s))
\end{array}$$