Cálculo de Programas

2.º Ano de LCC+MiEI (Universidade do Minho)
Ano Lectivo de 2019/20

Teste — 13 de Junho de 2020
10h00–13h00
Prova realizada on-line via BBC

- *Esta prova consta de **10** questões que valem, cada uma, 2.0 valores. O tempo médio estimado para resolução de cada questão é de 15 min.*
- *Os alunos devem indicar na sua prova o seu compromisso pessoal relativo à seguinte declaração:*

  Tendo presente o Código de Conduta Ética da Universidade do Minho e o Regulamento Disciplinar dos Estudantes da Universidade do Minho (Despacho RT-80/2019) e tendo conhecimento das sanções aplicáveis a atos de infração disciplinar, declaro por minha honra que pautarei a minha conduta na resolução desta prova de avaliação pelos valores de ética e integridade académica vigentes na Universidade do Minho. Declaro que realizarei a prova autonomamente e recorrendo exclusivamente aos elementos de consulta autorizada. Confirmo ainda que não incorrerei em qualquer ato de desonestidade, nomeadamente, os que violam os princípios éticos inerentes a processos de avaliação, como a prática de plágio ou qualquer outra forma indevida de utilização de informações. Mais declaro que não me envolverei em situações de prestação de informação ou apoio indevidos no decurso das provas que venha a realizar.
- *Nas perguntas de resposta por ficheiro, pode submeter um ficheiro de texto, uma imagem ou um zip com várias ficheiros.*

PROVA COM CONSULTA (3h)

Questão 1 ("dummy": na implementação em BBC era a declaração acima)

Questão 2 Recorde a função $\mathsf{undistr} : A \times B + A \times C \to A \times (B + C)$ definida como

$$[id \times i_1, id \times i_2] \qquad \text{(DEF)}$$

Esta função tem uma função inversa $\mathsf{distr} : A \times (B + C) \to A \times B + A \times C$, cuja definição é desconhecida para o contexto deste exercício. Mostre que

$$\mathsf{distr} \cdot (id \times \mathsf{coswap}) \cdot \mathsf{undistr} = \mathsf{coswap}$$

justificando os passos abaixo, sabendo que

$$\mathsf{coswap} = [i_2, i_1] \qquad \text{(DEF)}$$

Relembre que para isomorfismos f e g, temos que

$$\begin{cases} f \cdot h = k \equiv h = g \cdot k \\ h \cdot f = k \equiv h = k \cdot h \end{cases} \qquad \text{(ISO)}$$

Em cada caixa deve escrever exatamente um identificador da lei do cálculo tal como identificada no formulário da disciplina (ou dada no enunciado deste exercício), sem parenteses.

$$
\begin{array}{ll}
 & \mathsf{distr} \cdot (id \times \mathsf{coswap}) \cdot \mathsf{undistr} = \mathsf{coswap} \\
\equiv & \quad \{\ \text{[A]}\ \} \\
 & (id \times \mathsf{coswap}) \cdot \mathsf{undistr} = \mathsf{undistr} \cdot \mathsf{coswap} \\
\equiv & \quad \{\ \text{[B]}\ \} \\
 & [id \times i_1, id \times i_2] \cdot [i_2, i_1] \\
\equiv & \quad \{\ \text{[C]}\ \} \\
 & [[id \times i_1, id \times i_2] \cdot i_2, [id \times i_1, id \times i_2] \cdot i_1] \\
\equiv & \quad \{\ \text{[D]}\ \} \\
 & [id \times i_2, id \times i_1] \\
\equiv & \quad \{\ \text{[E]}\ \} \\
 & [id \times ([i_2, i_1] \cdot i_1), id \times ([i_2, i_1] \cdot i_2)] \\
\equiv & \quad \{\ \text{[F], 1}\ \} \\
 & [(id \times [i_2, i_1]) \cdot (id \times i_1), (id \times [i_2, i_1]) \cdot (id \times i_2)] \\
\equiv & \quad \{\ \text{[G]}\ \} \\
 & (id \times [i_2, i_1]) \cdot [id \times i_1, id \times i_2] \\
\equiv & \quad \{\ \text{[H]}\ \} \\
 & (id \times \mathsf{coswap}) \cdot \mathsf{undistr} \\
\Box &
\end{array}
$$

Questão 3 Na biblioteca Cp.hs aparece a função genérica [1] $lstr :: Functor\ f \Rightarrow (a, f\ b) \to f\ (a, b)$, isto é,

$$lstr : A \times \mathsf{F}\ B \to \mathsf{F}\ (A \times B)$$

onde F é um functor. Para o caso $\mathsf{F}\ X = X \times C$, para um dado C fixo, $lstr$ coincide com um isomorfismo que conhece, qual? Faça escolha múltipla entre:

swap ; assocr ; assocl ; undistl ; undistr ; coswap ; nenhum destes.

RESOLUÇÃO: Para $\mathsf{F}\ X = X \times C$ ter-se-á

$$lstr : A \times (B \times C) \to (A \times B) \times C$$

É imediato que neste caso $lstr$ coincide com assocl, cf.

$$\begin{array}{l} \mathsf{assocl} :: (a, (b, c)) \to ((a, b), c) \\ \mathsf{assocl} = \langle id \times \pi_1, \pi_2 \cdot \pi_2 \rangle \end{array}$$

em Cp.hs. □

---

[1] Apenas se dá uma das alternativas — as outras são em tudo semelhantes.

Questão 4 Considere a função [2]

- $\delta = [\mathsf{map}\ \pi_2\,, singl \cdot \pi_1]$

para $singl\ x = [x]$. Infira o tipo mais geral de δ e deduza a respectiva propriedade grátis, que deverá verificar analiticamente. Tenha em consideração a propriedade natural de $singl$.

RESOLUÇÃO: Num teste aberto, não presencial, a tarefa de inferir o tipo está muito simplificada: basta pedi-lo ao GHCi:

- `delta :: Either [(a, b1)] (b1, b2) -> [b1]`

Isto é,

- $\delta : (A \times B)^* + B \times C \rightarrow B^*$

A dedução da propriedade grátis, feita sobre o habitual diagrama, deverá dar

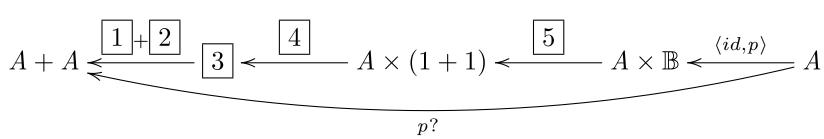
- $g^* \cdot \delta = \delta \cdot ((f \times g)^* + g \times h)$

A parte mais importante nesta questão é a prova analítica (completar com as justificações):

$$
\begin{array}{ll}
& g^* \cdot \delta = \delta \cdot ((f \times g)^* + g \times h) \\
\equiv & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& g^* \cdot [(\pi_2{}^*), singl \cdot \pi_1] = [(\pi_2{}^*), singl \cdot \pi_1] \cdot ((f \times g)^* + g \times h) \\
\equiv & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& [(g \cdot \pi_2)^*, g^* \cdot singl \cdot \pi_1] = [(\pi_2 \cdot (f \times g))^*, singl \cdot \pi_1 \cdot (g \times h)] \\
\equiv & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \left\{ \begin{array}{l} (g \cdot \pi_2)^* = (\pi_2 \cdot (f \times g))^* \\ g^* \cdot singl \cdot \pi_1 = singl \cdot \pi_1 \cdot (g \times h) \end{array} \right. \\
\equiv & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \left\{ \begin{array}{l} (g \cdot \pi_2)^* = (g \cdot \pi_2)^* \\ g^* \cdot singl \cdot \pi_1 = singl \cdot g \cdot \pi_1 \end{array} \right. \\
\equiv & \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& singl \cdot g \cdot \pi_1 = singl \cdot g \cdot \pi_1 \\
\equiv & \{ \ \} \\
& true
\end{array}
$$

□

Questão 5 Escolha uma das opções abaixo por forma a que a composição do diagrama descreva



(ainda que de forma complicada!) o guarda $p?$ de um dado predicado p (construção que se usa em condicionais de McCarthy):

[2] Apenas se dá uma das alternativas — as outras são em tudo semelhantes.

1. $1 = \pi_2; 2 = \pi_1; 3 = 1 \times A + A \times 1; 4 = \mathsf{distl}; 5 = id \times (p \to i_1, i_2)$
2. $1 = id; 2 = \pi_1; 3 = A + A \times 1; 4 = \mathsf{distr}; 5 = \langle id, f\ \rangle$ **where** $f\ \mathsf{True} = i_1\ (); f\ \mathsf{False} = i_2\ ()$
3. $1 = \pi_1; 2 = \pi_1; 3 = A \times 1 + A \times 1; 4 = \mathsf{distr}; 5 = \langle id, f\ \rangle$ **where** $f\ \mathsf{True} = i_1\ (); f\ \mathsf{False} = i_2\ ()$
4. Nenhuma das opções dadas tipa correctamente.

RESOLUÇÃO: Nenhuma das hipóteses está correcta. A que estaria era

$$A + A \xleftarrow{\pi_1+\pi_1} A \times 1 + A \times 1 \xleftarrow{\mathsf{distr}} A \times (1+1) \xleftarrow{id\times\alpha} A \times \mathbb{B} \xleftarrow{\langle id,p\rangle} A \qquad A \xrightarrow{p?} A + A$$

onde $\alpha^\circ = [true\,, false]$. □

Questão 6 A função $hasDup : A^* \to \mathbb{B}$ verifica se uma lista tem elementos duplicados. Por exemplo, $hasDup\ [\,] = \mathsf{False}$ e $hasDup\ [x, y, x] = \mathsf{True}$. A função pode ser implementada em Haskell da seguinte maneira:

$$\begin{array}{l} hasDup\ [\,] = \mathsf{False} \\ hasDup\ (x : xs) = elem\ (x, xs) \vee hasDup\ xs \end{array}$$

em que

- $elem : A \times A^* \to \mathbb{B}$ é uma função que verifica de um dado elemento está presente na lista, e
- $\vee : \mathbb{B} \times \mathbb{B} \to \mathbb{B}$ é a função Booleana "or".

Apresente a implementação desta função com base num catamorfismo de listas que tire partido da recursividade mútua. Para arrancar, pode começar por assumir que $hasDup = \pi_1 \cdot f$ e que

$$\begin{array}{l} f\ [\,] = \mathsf{False} \\ f\ (x : xs) = elem\ (x, g\ xs) \vee f\ xs \\ g\ [\,] = [\,] \\ g\ (x : xs) = x : g\ xs \end{array}$$

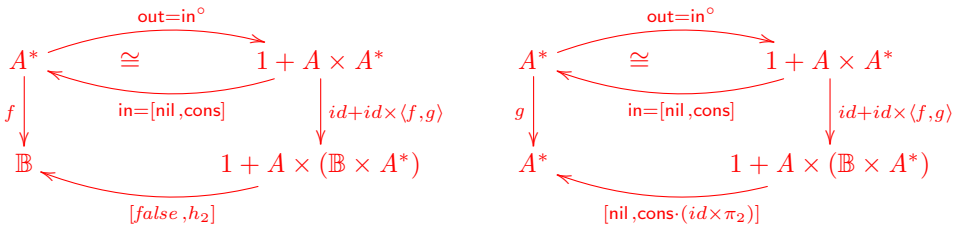
RESOLUÇÃO: Este exercício é em tudo semelhante ao caso de *ord* que foi apresentado nas aulas teóricas (cf. aula 7b, t=7:45). Segundo essa estratégia, começa-se pelos diagramas de f e de g, esta fácil de identificar como $(\!|\,\mathsf{in}\,|\!)$:

$$\begin{array}{ccc} A^* & \underset{\mathsf{in}=[\mathsf{nil},\mathsf{cons}]}{\overset{\mathsf{out}=\mathsf{in}^\circ}{\cong}} & 1 + A \times A^* \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle id+id\times\langle f,g\rangle} \\ \mathbb{B} & \xleftarrow{[false\,,h_2]} & 1 + A \times (\mathbb{B} \times A^*) \end{array} \qquad \begin{array}{ccc} A^* & \underset{\mathsf{in}=[\mathsf{nil},\mathsf{cons}]}{\overset{\mathsf{out}=\mathsf{in}^\circ}{\cong}} & 1 + A \times A^* \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle id+id\times g} \\ A^* & \xleftarrow{[\mathsf{nil},\mathsf{cons}]} & 1 + A \times A^* \end{array}$$

A função

$$h_2\ (x, (b, xs)) = elem\ (x, xs) \vee b$$

"decalca"-se a partir da cláusula $f\ (x : xs) = \ldots$. O passo seguinte é introduzir (falsa) recursividade mútua em g:



Ter-se-á então, pela lei de recursividade mútua,

$$\langle f, g\rangle = (\!|\, \langle [false\,, h_2], [\mathsf{nil}\,, \mathsf{cons} \cdot (id \times \pi_2)]\rangle \,|\!)$$

e não era pedido mais nada. Para quem tiver querido apresentar h_2 pointfree:

$$h_2 = \widehat{\nabla} \cdot \langle elem \cdot (id \times \pi_2), \pi_1 \cdot \pi_2\rangle$$

equivalente a

$$h_2 = \widehat{\nabla} \cdot (elem \times \pi_1) \cdot \langle id \times \pi_2, \pi_2\rangle$$

Notar o tipo de $\langle id \times \pi_2, \pi_2\rangle$ neste contexto: $A \times (\mathbb{B} \times A^*) \to (A \times A^*) \times (\mathbb{B} \times A^*)$. □

Questão 7 Considere os tipos indutivos Rose A (rose trees) e A^+ (listas não vazias) cujos bifunctores de base são, respectivamente, $\mathsf{B}\ (X,\ Y) = X \times Y^*$ e $\mathsf{B}\ (X,\ Y) = X + X \times Y$, e cujas álgebras de construção são, respectivamente, $inR = \mathsf{Rose}$ e $\mathsf{in} = [singl, \mathsf{cons}]$. É sempre possível converter de A^+ para Rose A através de um catamorfismo de A^+ com a forma $l2r = (\!|\, \alpha \cdot \beta \,|\!)$. Identifique o par (α, β) que faz correctamente essa conversão, por escolha múltipla entre:

1. $\alpha = \mathsf{Rose}$; $\beta = [id, \mathsf{map\ cons}]$
2. $\alpha = [\mathsf{nil}, \mathsf{cons}]$; $\beta = [\langle id, \mathsf{nil}\rangle, id \times singl]$
3. $\alpha = \mathsf{Rose}$; $\beta = \langle [id, \pi_1], [\mathsf{nil}, singl \cdot \pi_2]\rangle$
4. $\alpha = \mathsf{Rose}$; $\beta = [\langle id, \mathsf{nil}\rangle, \pi_1 \times singl]$
5. Nenhuma das opções acima.

RESOLUÇÃO: Se queremos Rose trees na saída, deveremos construir $(\!|\, inR \cdot \beta \,|\!)$ onde β tem de mapear o bifunctor das listas no das Rose trees, $\beta : X + X \times Y \to X \times Y^*$. Fazendo $\beta = [f\,, g]$ vemos que

$$f : X \to X \times Y^*$$
$$f = \langle id, \mathsf{nil}\rangle$$
$$g : X \times Y \to X \times Y^*$$
$$g = id \times singl$$

são as escolhas a fazer. □

Questão 8 Considere o tipo de dados $\mathbf{data}\ \mathsf{Rose}\ a = \mathsf{Rose}\ (a, [\mathsf{Rose}\ a])$ que representa árvores cujos nós são do tipo a e que contêm um número arbitrário de filhos (rose trees). Considere também a seguinte função,

$$roseMap :: (a \to b) \to \mathsf{Rose}\ a \to \mathsf{Rose}\ b$$
$$roseMap\ f\ (\mathsf{Rose}\ (x, s)) = \mathsf{Rose}\ (f\ x, \mathsf{map}\ (roseMap\ f)\ s)$$

Finalmente, considere a função

$$depth :: \mathsf{Rose}\ a \to \mathbb{Z}$$
$$depth = (\!|\ \mathsf{succ} \cdot maximum \cdot \pi_2\ |\!)$$

em que $maximum :: [\mathbb{Z}] \to \mathbb{Z}$ é uma função que calcula o maior elemento de uma dada lista, assumindo que $maximum\ [\,] = 0$. (A função $depth$ é um catamorfismo em Rose que calcula a profundidade máxima de uma dada rose tree.)

O objetivo deste exercício é provar que a igualdade

$$depth = depth \cdot (roseMap\ f)$$

é verdadeira. Para tal, primeiro deve provar que $roseMap\ f = (\!|\ \mathsf{in} \cdot \mathsf{B}\ (f, id)\ |\!)$ para uma certa função in e para um certo bifunctor B. De seguida, deve provar a igualdade pretendida.

RESOLUÇÃO: Primeiro há que ideintificar B e in em $roseMap\ f = (\!|\ \mathsf{in} \cdot \mathsf{B}\ (f, id)\ |\!)$ — completar com as justificações omitidas:

$$roseMap\ f\ (\mathsf{Rose}\ (x, s)) = \mathsf{Rose}\ (f\ x, \mathsf{map}\ (roseMap\ f)\ s)$$
$$\equiv \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \}$$
$$(roseMap\ f \cdot \mathsf{Rose})\ (x, s) = (\mathsf{Rose} \cdot (f \times \mathsf{map}\ (roseMap\ f))\ (x, s)$$
$$\equiv \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \}$$
$$roseMap\ f \cdot \mathsf{in} = \mathsf{in} \cdot (f \times id) \cdot (id \times \mathsf{map}\ (roseMap\ f))$$
$$\equiv \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \}$$
$$roseMap\ f \cdot \mathsf{in} = \mathsf{in} \cdot \mathsf{B}\ (f, id) \cdot \mathsf{B}\ (id, roseMap\ f)\ \mathbf{where}\ \mathsf{B}\ (f, g) = f \times \mathsf{map}\ g$$
$$\equiv \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \}$$
$$roseMap\ f = (\!|\ \mathsf{in} \cdot \mathsf{B}\ (f, id)\ |\!)\ \mathbf{where}\ \mathsf{B}\ (f, g) = f \times \mathsf{map}\ g$$

Então:

$$\equiv \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \}$$
$$depth = depth \cdot (\!|\ \mathsf{in} \cdot \mathsf{B}\ (f, id)\ |\!)\ \mathbf{where}\ \mathsf{B}\ (f, g) = f \times \mathsf{map}\ g$$
$$\equiv \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \}$$
$$depth = (\!|\ \mathsf{succ} \cdot maximum \cdot \pi_2 \cdot (f \times id)\ |\!)$$
$$\equiv \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \}$$
$$depth = (\!|\ \mathsf{succ} \cdot maximum \cdot \pi_2\ |\!)$$
$$\equiv \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \}$$
$$true$$

□

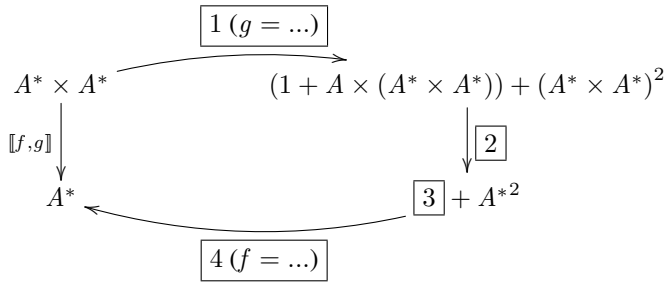
□

Questão 9 Pretende-se exprimir com recurso a um hilomorfismo a função $lcs : A^* \times A^* \to A^*$ que calcula a mais longa subsequência comum entre duas listas (lcs = 'longest common subsequence'):

$$lcs\ [\,]\ \_ = [\,]$$
$$lcs\ \_\ [\,] = [\,]$$

$$
\begin{array}{l}
lcs\ (x:xs)\ (y:ys) = \\
\quad \textbf{if}\ x \equiv y\ \textbf{then}\ x : lcs\ xs\ ys\ \textbf{else} \\
\quad\quad \textbf{let}\ a = lcs\ xs\ (y:ys) \\
\quad\quad\quad\ b = lcs\ (x:xs)\ ys \\
\quad\quad \textbf{in if}\ \mathsf{length}\ a > \mathsf{length}\ b\ \textbf{then}\ a\ \textbf{else}\ b
\end{array}
$$

Por exemplo, lcs `"abcd" "aacbed" = "abd"`. Preencha as 4 caixas numeradas com a informação em falta por forma a que $lcs = \overline{[\![f, g]\!]}$:



RESOLUÇÃO: NB: a resolução que se propõe é mais detalhada do que era preciso, aproveitando-se para explicar melhor a construção de hilomorfismos em Cálculo de Programas.[3]

Em primeiro lugar, há que inferir o functor de base B, com base no parâmetro A e na recursividade feita sobre $A^* \times A^*$. Seguindo o método que se estudou, há que substituir $A^* \times A^*$ por Y em $(1 + A \times (A^* \times A^*)) + (A^* \times A^*)^2$ obtendo-se $(1 + A \times Y) + Y^2$. Finalmente, substitui-se A por X e obtém-se:

$$
\begin{array}{l}
\mathsf{B}\ (X, Y) = (1 + X \times Y) + Y^2 \\
\mathsf{B}\ (f, g) = (id + f \times g) + g^2
\end{array}
$$

Daqui tira-se $\mathsf{F}\ g = \mathsf{B}\ (id, g) = (id + id \times g) + g^2$. Logo:

$$
\begin{array}{l}
\boxed{2} = (id + id \times [\![f, g]\!]) + [\![f, g]\!]^2 \\
\boxed{3} = (1 + A \times A^*) + A^{*2}
\end{array}
$$

Como estamos em listas na "parte cata", para $\boxed{4}$ infere-se $f = [\mathsf{in}, f_2]$. Para obtermos f_2 temos que ir ver o que o algoritmo faz às saídas a e b das chamadas recursivas, nessa posição:

$$
f_2\ (a, b) = \textbf{if}\ \mathsf{length}\ a > \mathsf{length}\ b\ \textbf{then}\ a\ \textbf{else}\ b
$$

A caixa $\boxed{1}$ é a menos imediata, pois neste hilomorfismo a maior parte das decisões algorítmicas têm lugar na "parte ana". Como $[\![f, g]\!] = \widehat{lcs}$, podemos aplicar *uncurry* ao código dado, onde h abrevia $[\![f, g]\!] = \widehat{lcs}$, obtendo-se:

$$
\begin{array}{l}
h\ ([\,], \_) = [\,] \\
h\ (\_, [\,]) = [\,] \\
h\ ((x:xs)\ (y:ys) = \\
\quad \textbf{if}\ x \equiv y\ \textbf{then}\ x : h\ (xs, ys)\ \textbf{else} \\
\quad\quad \textbf{let}\ a = h\ (xs, (y:ys)) \\
\quad\quad\quad\ b = h\ ((x:xs), ys) \\
\quad\quad \textbf{in if}\ \mathsf{length}\ a > \mathsf{length}\ b\ \textbf{then}\ a\ \textbf{else}\ b
\end{array}
$$

O que o gene da caixa $\boxed{1}$ faz é preparar as chamadas recursivas de h. A lista vazia $[\,]$ e $(:)$ têm a ver com a saída, cf. in da caixa $\boxed{4}$. Olhando para F, onde há uma alternativa dentro de outra, definimos

$$
\begin{array}{l}
i_{11} = i_1 \cdot i_1 \\
i_{12} = i_1 \cdot i_2
\end{array}
$$

na seguinte formulação de g:

[3] Comparar também com a experiência adquirida em determinadas funções pedidas no trabalho prático.

$$g\ ([\,], \_) = i_{11}\ ()$$
$$g\ (\_, [\,]) = i_{11}\ ()$$
$$g\ (x : xs, y : ys) = \textbf{if}\ x \equiv y\ \textbf{then}\ i_{12}\ (x, (xs, ys))\ \textbf{else}\ i_2\ ((xs, y : ys), (x : xs, ys))$$

□

Questão 10 Considere uma função $g : X \to X$. Construa como um catamorfismo de números naturais a função $manyg : \mathbb{N}_0 \to X \to X$ tal que, dado um número natural n e um valor x, $manyg\ n\ x$ retorna a aplicação de g ao valor x, n vezes. Por exemplo, $manyg\ 2\ x = g\ (g\ (x))$ e $manyg\ 0\ x = x$.

Note que $f = \widehat{manyg}$ é a função seguinte:

$$f\ (0, x) = x$$
$$f\ (n + 1, x) = g\ (f\ (n, x))$$

RESOLUÇÃO: Vamos partir da função f dada (completar com justificações):

$$\left\{ \begin{array}{l} f\ (0, x) = x \\ f\ (n + 1, x) = g\ (f\ (n, x)) \end{array} \right.$$
$$\equiv \quad \{\ \text{pointwise currying}\ \}$$
$$\left\{ \begin{array}{l} \overline{f}\ 0\ x = x \\ \overline{f}\ (n + 1)\ x = g\ (\overline{f}\ n\ x) \end{array} \right.$$
$$\equiv \quad \{\ \ldots\ \}$$
$$\left\{ \begin{array}{l} (\overline{f}\ 0)\ x = id\ x \\ (\overline{f}\ (n + 1))\ x = g\ ((\overline{f}\ n)\ x) \end{array} \right.$$
$$\equiv \quad \{\ \ldots\ \}$$
$$\left\{ \begin{array}{l} \overline{f}\ 0 = id \\ \overline{f}\ (n + 1) = (g^X \cdot \overline{f})\ n \end{array} \right.$$
$$\equiv \quad \{\ \ldots\ \}$$
$$\overline{f} \cdot [\mathsf{zero}, \mathsf{succ}] = [\underline{id}, g^X] \cdot (id + \overline{f})$$
$$\equiv \quad \{\ \ldots\ \}$$
$$\overline{f} = (\!|\, [\underline{id}, g^X] \,|\!)$$

□

Ou seja, $\overline{f} = \mathsf{for}\ g^X\ id$. □

Questão 11 A função *discollect* que foi assunto de várias questões das fichas práticas tem uma definição monádica genérica,

$$discollect = lstr \bullet id \qquad \text{(DIS)}$$

onde *lstr* é uma função que encontra no módulo Cp.hs e que satisfaz a propriedade

$$lstr \cdot (id \times u) = u \qquad \text{(PROP1)}$$

onde u é a unidade do mónade onde *discollect* opera. Pretende-se demonstrar a igualdade:

$$discollect \cdot u \cdot (id \times u) = u \qquad \text{(PROP2)}$$

Comece por instanciar a propriedade (PROP2) no mónade LTree, em notação pointwise. Caso necessário, utilize nomes de variáveis a, b, c, ...

[A] De seguida complete as justificações da prova de (PROP2) com leis que conhece do formulário da disciplina ou com factos dados acima. Em cada caixa deve escrever exatamente um identificador da lei do cálculo tal como identificada no formulário da disciplina (ou dada no enunciado deste exercício), sem parenteses.

$$\begin{array}{ll} & discollect \cdot u \cdot (id \times u) = u \\ \equiv & \quad \{ \text{ [B] } \} \\ & (lstr \bullet id) \cdot u \cdot (id \times u) = u \\ \equiv & \quad \{ \text{ [C] } \} \\ & (lstr \bullet (id \cdot u)) \cdot (id \times u) = u \\ \equiv & \quad \{ \text{ [D] } \} \\ & (lstr \bullet u) \cdot (id \times u) = u \\ \equiv & \quad \{ \text{ [E] } \} \\ & lstr \cdot (id \times u) = u \\ \equiv & \quad \{ \text{ [F] } \} \\ & true \end{array}$$

□

RESOLUÇÃO: Estudou-se que LTree forma o mónade

$$X \xrightarrow{Leaf} \mathsf{LTree}\ X \xleftarrow{(\!|\, [id, Fork] \,|\!)} \mathsf{LTree}^2\ X$$

– cf. e.g. vídeos da aulas teóricas, aula 11b, t=20:05. Logo PROP2 instancia neste mónade como se segue:

$$\begin{array}{ll} & discollect \cdot u \cdot (id \times u) = u \\ \equiv & \quad \{ \ u = Leaf \text{ neste mónade } \} \\ & discollect \cdot Leaf \cdot (id \times Leaf) = Leaf \\ \equiv & \quad \{ \text{ introdução de variáveis (etc) } \} \\ & discollect\ (Leaf\ (a, Leaf\ b) = Leaf\ (a, b) \end{array}$$

□
