

Métodos de Programação I

2.º Ano da LMCC (701055) + LESI (531316)
Ano Lectivo de 2000/2001

Exame (época de recurso) — 3 de Setembro de 2001
14h30
Salas 2203 a 2206

NB: Esta prova consta de **10** alíneas que valem, cada uma, 2 valores.

PROVA SEM CONSULTA (3 horas)

Questão 1 Partindo da definição do “combinador condicional” de McCarthy e da propriedade

$$p? \cdot f = (f + f) \cdot (p \cdot f)? \quad (1)$$

prove a validade de

$$(p \rightarrow f, g) \cdot h = (p \cdot h) \rightarrow (f \cdot h), (g \cdot h) \quad (2)$$

RESOLUÇÃO:

$$\begin{aligned} (p \rightarrow f, g) \cdot h & \\ & \equiv \{ \text{definição do combinador condicional de McCarthy} \} \\ & ([f, g] \cdot p?) \cdot h \\ & \equiv \{ \text{composição é associativa} \} \\ & [f, g] \cdot (p? \cdot h) \\ & \equiv \{ \text{propriedade (1)} \} \\ & [f, g] \cdot (h + h) \cdot (p \cdot h)? \\ & \equiv \{ \text{absorção-+ (17)} \} \\ & [f \cdot h, g \cdot h] \cdot (p \cdot h)? \\ & \equiv \{ \text{definição do combinador condicional de McCarthy} \} \\ & (p \cdot h) \rightarrow (f \cdot h), (g \cdot h) \end{aligned}$$

□

Questão 2 Dadas as definições

$$\begin{aligned} f &= (h \cdot \pi_2) \cdot swap \\ g &= \pi_1 \cdot (h \times (id^A \cdot ap)) \end{aligned}$$

represente f e g sob a forma de diagramas evidenciando o seu tipo, e mostre que $f = g$, para todo o h .

RESOLUÇÃO: Partindo de $C \xleftarrow{h} B$, infere-se $C \xleftarrow{f} B \times D$, cf. o diagrama

$$\begin{array}{ccc} B \times D & & \\ \downarrow \text{swap} & & \\ D \times B & \xrightarrow{\pi_2} & B \\ & & \downarrow h \\ & & C \end{array}$$

e $C \xleftarrow{g} B \times ((F^A)^E \times E)$, cf. o diagrama

$$\begin{array}{ccc} B \times ((F^A)^E \times E) & & \\ \downarrow h \times ap & & \\ C \times (F^A) & & \\ \downarrow id \times id^A & & \\ C & \xleftarrow{\pi_1} & C \times F^A \end{array}$$

Os dois tipos unificam para $D = (F^A)^E \times E$, o que abre caminho à prova da igualdade:

$$\begin{aligned} f &= g \\ \equiv & \quad \{ \text{substituição} \} \\ & (h \cdot \pi_2) \cdot \text{swap} = \pi_1 \cdot (h \times (id^A \cdot ap)) \\ \equiv & \quad \{ \text{composição é associativa, definição de swap e (26)} \} \\ & h \cdot (\pi_2 \cdot \langle \pi_2, \pi_1 \rangle) = \pi_1 \cdot (h \times ap) \\ \equiv & \quad \{ \text{cancelamento-} \times \text{ (7) e } \pi_1 \text{ é natural} \} \\ & h \cdot \pi_1 = h \cdot \pi_1 \\ \equiv & \quad \{ \text{igualdade é reflexiva} \} \\ & V \end{aligned}$$

□

Questão 3 Na programação funcional é vulgar a ocorrência de funções parciais, *i.é.* funções indefinidas para algum dos seus argumentos. Por exemplo, a divisão é parcial pois $n/0$ é um valor indefinido, ou *excepção*. As exceções são vulgarmente assinaladas através de mensagens de erro, estendendo-se o codomínio da função por forma a fornecer ‘strings’ explicativos. Em HASKELL, por exemplo,

```
(/) :: Double -> Double -> Double
```

pode ser estendida a

```
dv :: (Double, Double) -> Error Double
dv(n,0) = Err "Nem pense em dividir por 0!"
dv(n,m) = Ok (n / m)
```

onde

```
data Error a = Err String | Ok a deriving Show
```

Outro exemplo ocorre no processamento de listas

```
hd [] = Err "Lista vazia!"
hd (a:_) = Ok a
```

Para evitar a proliferação arbitrária de condições de teste de exceções e seu processamento pode definir-se um combinador de composição de funções parciais da forma seguinte:

```
(.!) :: (a -> Error b) -> (c -> Error a) -> c -> Error b
g .! f = errh . (fmap g) . f
```

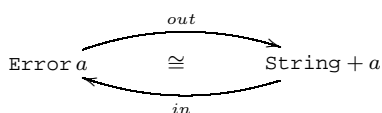
assumindo um combinador de exceções `errh` (=‘error handler’)

```
errh (Err e) = Err e
errh (Ok a) = a
```

e

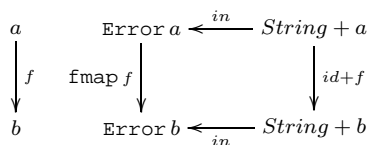
```
instance Functor Error where
  fmap f (Ok a) = Ok (f a)
  fmap f (Err e) = Err e
```

1. Qual o tipo de `errh`? Escreva a mesma função em notação sem variáveis, acompanhada de um diagrama explicativo, com base no isomorfismo



onde $\text{in} = [\text{Err}, \text{Ok}]$.

2. Calcule a definição em HASKELL de `fmap` a partir da interpretação do diagrama que se segue:



3. Preencha as reticências “...A...” a “...F...” na seguinte prova de functorialidade de `fmap`:

$$\begin{aligned} & \text{fmap } (f \cdot g) = (\text{fmap } f) \cdot (\text{fmap } g) \\ \equiv & \quad \{ \dots A \dots \} \\ & (\text{fmap } (f \cdot g)) \cdot \text{in} = (\text{fmap } f) \cdot (\text{fmap } g) \cdot \text{in} \\ \equiv & \quad \{ \dots B \dots \} \\ & (\text{fmap } (f \cdot g)) \cdot \text{in} = (\text{fmap } f) \cdot \text{in} \cdot (\text{id} + g) \\ \equiv & \quad \{ \dots C \dots \} \\ & (\text{fmap } (f \cdot g)) \cdot \text{in} = \text{in} \cdot (\text{id} + f) \cdot (\text{id} + g) \\ \equiv & \quad \{ \dots D \dots \} \\ & (\text{fmap } (f \cdot g)) \cdot \text{in} = \text{in} \cdot (\text{id} + f \cdot g) \\ \equiv & \quad \{ \dots E \dots \} \\ & (\text{fmap } (f \cdot g)) \cdot \text{in} = (\text{fmap } (f \cdot g)) \cdot \text{in} \\ \equiv & \quad \{ \dots F \dots \} \\ & V \end{aligned}$$

RESOLUÇÃO:

1. Na definição de `(.!)` tem-se

$$\text{Error } a \xleftarrow{f} c$$

e

$$\text{Error } b \xleftarrow{g} a$$

logo tem-se

$$\text{Error}(\text{Error } b) \xleftarrow{\text{fmap } g} \text{Error } a$$

e

$$\text{Error}(\text{Error } b) \xleftarrow{\text{fmap } g \cdot f} c$$

Como

$$\text{Error } b \xleftarrow{g \cdot !f} c$$

terá que ter o mesmo tipo que $\text{errh} \cdot \text{fmap } g \cdot f$, conclui-se

$$\text{Error } b \xleftarrow{\text{errh}} \text{Error}(\text{Error } b)$$

Conversão para notação sem variáveis:

$$\begin{aligned} & \left\{ \begin{array}{l} \text{errh}(\text{Err } e) = \text{Err } e \\ \text{errh}(\text{Ok } a) = a \end{array} \right. \\ \equiv & \quad \{ \text{composição de funções e introdução da identidade} \} \\ & \left\{ \begin{array}{l} (\text{errh} \cdot \text{Err})e = \text{Err } e \\ (\text{errh} \cdot \text{Ok})a = \text{id } a \end{array} \right. \\ \equiv & \quad \{ \text{remoção de variáveis} \} \\ & \left\{ \begin{array}{l} \text{errh} \cdot \text{Err} = \text{Err} \\ \text{errh} \cdot \text{Ok} = \text{id} \end{array} \right. \\ \equiv & \quad \{ \text{junção das equações via "either"} \} \\ & [\text{errh} \cdot \text{Err}, \text{errh} \cdot \text{Ok}] = [\text{Err}, \text{id}] \\ \equiv & \quad \{ \text{fusão-+ (16) em sentido inverso e definição de in} \} \\ & \text{errh} \cdot \text{in} = [\text{Err}, \text{id}] \end{aligned}$$

2. Tem-se:

$$\begin{aligned} & \text{fmap } f \cdot \text{in} = \text{in} \cdot (\text{id} + f) \\ \equiv & \quad \{ \text{definição de in} \} \\ & \text{fmap } f \cdot [\text{Err}, \text{Ok}] = [\text{Err}, \text{Ok}] \cdot (\text{id} + f) \\ \equiv & \quad \{ \text{fusão-+ (16), absorção-+ (17) e (4)} \} \\ & [\text{fmap } f \cdot \text{Err}, \text{fmap } f \cdot \text{Ok}] = [\text{Err}, \text{Ok} \cdot f] \\ \equiv & \quad \{ \text{igualdade estrutural de "eithers"} \} \\ & \left\{ \begin{array}{l} \text{fmap } f \cdot \text{Err} = \text{Err} \\ \text{fmap } f \cdot \text{Ok} = \text{Ok} \cdot f \end{array} \right. \\ \equiv & \quad \{ \text{introdução de variáveis} \} \\ & \left\{ \begin{array}{l} \text{fmap } f(\text{Err } e) = \text{Err } e \\ \text{fmap } f(\text{Ok } a) = \text{Ok}(f a) \end{array} \right. \end{aligned}$$

3. Tem-se:

$$\begin{aligned} & \text{fmap } (f \cdot g) = (\text{fmap } f) \cdot (\text{fmap } g) \\ \equiv & \quad \{ \text{in é isomorfismo} \} \\ & (\text{fmap } (f \cdot g)) \cdot \text{in} = (\text{fmap } f) \cdot (\text{fmap } g) \cdot \text{in} \\ \equiv & \quad \{ \text{diagrama da alínea anterior} \} \\ & (\text{fmap } (f \cdot g)) \cdot \text{in} = (\text{fmap } f) \cdot \text{in} \cdot (\text{id} + g) \\ \equiv & \quad \{ \text{diagrama da alínea anterior} \} \end{aligned}$$

$$\begin{aligned}
& (\text{fmap } (f \cdot g)) \cdot \text{in} = \text{in} \cdot (\text{id} + f) \cdot (\text{id} + g) \\
\equiv & \quad \{ \text{leis functor-+ (18) e (4)} \} \\
& (\text{fmap } (f \cdot g)) \cdot \text{in} = \text{in} \cdot (\text{id} + f \cdot g) \\
\equiv & \quad \{ \text{diagrama da alínea anterior} \} \\
& (\text{fmap } (f \cdot g)) \cdot \text{in} = (\text{fmap } (f \cdot g)) \cdot \text{in} \\
\equiv & \quad \{ \text{igualdade é reflexiva} \} \\
& V
\end{aligned}$$

□

Questão 4 Antes de resolver as duas alíneas desta questão analize com atenção a seguinte arquitectura para a função

$$\text{mdc} \stackrel{\text{def}}{=} \text{mul} \cdot \text{fpc} \cdot (\text{fp} \times \text{fp})$$

que calcula o máximo divisor comum entre dois números naturais, cf. o diagrama

$$\mathbb{N} \times \mathbb{N} \xrightarrow{\text{fp} \times \text{fp}} \mathbb{N}^* \times \mathbb{N}^* \xrightarrow{\text{fpc}} \mathbb{N}^* \xrightarrow{\text{mul}} \mathbb{N}$$

onde

- fp calcula os factores primos de um número listados por ordem crescente, e.g. $\text{fp } 60 = [2, 2, 3, 5]$ e $\text{fp } 42 = [2, 3, 7]$;
- fpc intersecta duas listas de factores primos, e.g. $\text{fpc}([2, 2, 3, 5], [2, 3, 7]) = [2, 3]$ (fpc abrevia “factores primos comuns”);
- mul multiplica os factores da lista produzida por fpc , inferindo assim o máximo divisor comum — cf. $\text{mdc}(60, 42) = 2 * 3 = 6$.

1. Complete a seguinte definição, em HASKELL, da função

```

fpc' [] r = .....
fpc' l [] = .....
fpc' (a:l) (b:r) | a == b = .....
                  | a <  b = .....
                  | a >  b = .....

```

que é a versão “curried” de fpc (i.e. $\text{fpc}' = \overline{\text{fpc}}$):

2. Escreva fpc como um hilomorfismo e mul como um catamorfismo (de listas).

RESOLUÇÃO:

1. Tem-se:

```

fpc' [] r = []
fpc' l [] = []
fpc' (a:l) (b:r) | a == b = a : (fpc' l r)
                  | a <  b = fpc' l (b:r)
                  | a >  b = fpc' (a:l) r

```

2. mul é o catamorfismo bem conhecido que multiplica todos os elementos de uma sequência: $\text{mul} = \llbracket \underline{1}, * \rrbracket$.

Exprimir fpc como um hilomorfismo de listas significa encontrar A , g e h do diagrama que se segue:

$$\begin{array}{ccc}
\mathbb{N}^* & \xleftarrow{g} & 1 + A \times \mathbb{N}^* \\
\uparrow \llbracket g \rrbracket & & \uparrow \text{id} + \text{id} \times \llbracket h \rrbracket \\
A^* & \xleftarrow{\text{in}} & 1 + A \times A^* \\
\uparrow \llbracket h \rrbracket & & \uparrow \text{id} + \text{id} \times \llbracket h \rrbracket \\
\mathbb{N}^* \times \mathbb{N}^* & \xrightarrow{h} & 1 + A \times (\mathbb{N}^* \times \mathbb{N}^*)
\end{array}$$

A curved arrow labeled fpc points from $\mathbb{N}^* \times \mathbb{N}^*$ to A^* .

Devemos pensar em encontrar A primeiro que tudo. As três primeiras cláusulas de fpc' sugerem $A = \mathbb{N}$, mas para $a < b$ e $a > b$ não existe nenhum valor do tipo A para construir a lista A^* intermédia, o que sugere $A = 1 + \mathbb{N}$. Assim, o diagrama fica:

$$\begin{array}{ccc}
 \mathbb{N}^* & \xleftarrow{g} & 1 + (1 + \mathbb{N}) \times \mathbb{N}^* \\
 \uparrow \llbracket g \rrbracket & & \uparrow id + id \times \llbracket h \rrbracket \\
 fpc(1 + \mathbb{N})^* & \xleftarrow{in} & 1 + (1 + \mathbb{N}) \times (1 + \mathbb{N})^* \\
 \uparrow \llbracket h \rrbracket & & \uparrow id + id \times \llbracket h \rrbracket \\
 \mathbb{N}^* \times \mathbb{N}^* & \xrightarrow{h} & 1 + (1 + \mathbb{N}) \times (\mathbb{N}^* \times \mathbb{N}^*)
 \end{array}$$

Podemos agora decalcar h a partir de fpc' , usando Maybe A para representar $1 + A$:

```

h([],r) = i1()
h(1,[],) = i1()
h(a:l,b:r) | a == b = i2(Just a,(l,r))
             | a < b = i2(Nothing,(l,b:r))
             | a > b = i2(Nothing,(a:l,r))

```

Quanto a g , será sempre da forma $\llbracket _, g' \rrbracket$ onde g' apenas tem que ignorar os valores nulos (Nothing):

```

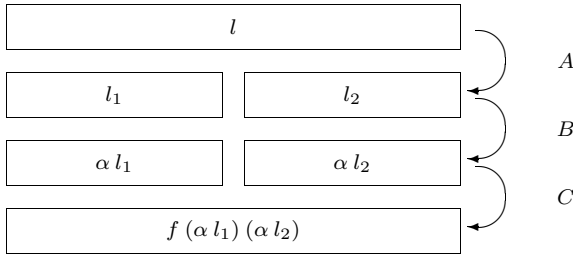
g'(Just a,l) = a:l
g'(Nothing,l) = l

```

Se recorrermos ao módulo RList, teremos `fpc = hyloRList (either (const []) g') h`.

□

Questão 5



O desenho ao lado pretende descrever graficamente um algoritmo de ordenação $A^* \xrightarrow{\alpha} A^*$ que conhece:

1. Identifique α , bem como as suas fases A, B e C e a função f . Codifique esta última em HASKELL.
2. Descreva o mesmo algoritmo sob a forma de um hilomorfismo de um particular tipo indutivo estudado nas aulas desta disciplina.

RESOLUÇÃO:

1. $\alpha = \text{mSort}$ ('merge sort') e f é a função merge da biblioteca `LTree.hs`. Fases do algoritmo: A — análise da lista argumento e sua representação em duas sublistas de igual (ou quase igual) comprimento; B — passo recursivo, i.e ordenação das duas sublistas construídas em A ; C — síntese do resultado por fusão de duas listas ordenadas.

2. Ver `mSort` em `LTree.hs`.

O hilomorfismo implícito em 'mSort' exclui listas não-vazias, que estão trivialmente ordenadas. Listas singulares também estão trivialmente ordenadas e como tal não oferecem problemas. Na fase A , uma função $A^* \xrightarrow{sep} A^* \times A^*$ vai partir listas não singulares em pares de listas não vazias, sem 'pivot' (é esta a principal diferença entre o 'quick sort' e o 'merge sort'). Isto sugere, em conjunto com o caso singular, a seguinte estrutura de dados intermédia para o hilomorfismo:

$$LTree\ A \cong A + LTree\ A \times LTree\ A$$

conhecida pelo nome de *árvore com folhas*. Surge assim o anamorfismo "de separação"

$$\begin{array}{ccc}
 LTree\ A & \xleftarrow{in} & A + LTree\ A \times LTree\ A \\
 \uparrow \llbracket h \rrbracket & & \uparrow id + \llbracket h \rrbracket \times \llbracket h \rrbracket \\
 A^* & \xrightarrow{h} & A + A^* \times A^*
 \end{array}$$

onde h (designada *lsplit* em `LTree.hs`) tem o propósito de bilinearizar o algoritmo (tal como o seu equivalente no ‘quick sort’). A estrutura de dados intermédia será agora consumida por um catamorfismo baseado na função $A^* \xleftarrow{\text{merge}} A^* \times A^*$ que funde listas ordenadas. Basta construir $\llbracket g \rrbracket$ para $g = [\text{singl}, \text{merge}]$ e adicioná-lo ao diagrama:

$$\begin{array}{ccc}
 A^* & \xleftarrow{g} & A + A^* \times A^* \\
 \uparrow \llbracket g \rrbracket & & \uparrow id + \llbracket g \rrbracket \times \llbracket g \rrbracket \\
 \text{LTree } A & \xleftarrow{\text{in}} & A + \text{LTree } A \times \text{LTree } A \\
 \uparrow \llbracket h \rrbracket & & \uparrow id + \llbracket h \rrbracket \times \llbracket h \rrbracket \\
 A^* & \xrightarrow{h} & A + A^* \times A^*
 \end{array}$$

□

Questão 6 A seguinte versão linear do algoritmo de Fibonacci,

```

fib n = snd (f n)

f 0      = (0,1)
f (n+1) = let (a,b) = f n
           in (b,a+b)

```

é uma codificação em HASKELL cuja função auxiliar f é o catamorfismo de naturais representado no diagrama que se segue:

$$\begin{array}{ccc}
 \mathbb{N} & \xleftarrow{[0, \text{succ}]} & 1 + \mathbb{N} \\
 f \downarrow & & \downarrow id + f \\
 \mathbb{N} \times \mathbb{N} & \xleftarrow{g} & 1 + \mathbb{N} \times \mathbb{N}
 \end{array} \tag{3}$$

Caracterize o “gene” g do catamorfismo em causa.

RESOLUÇÃO: Começa-se por re-escrever f sem variáveis:

$$\begin{cases} f \cdot \underline{0} = \langle 0, 1 \rangle \\ f \cdot \text{succ} = \langle \pi_2, + \rangle \cdot f \end{cases}$$

Daqui resulta

$$\begin{aligned}
 f \cdot [0, \text{succ}] &= [\langle 0, 1 \rangle, \langle \pi_2, + \rangle \cdot f] \\
 \equiv & \quad \{ \text{absorção-+ (17) em sentido inverso} \} \\
 f \cdot [0, \text{succ}] &= [\langle 0, 1 \rangle, \langle \pi_2, + \rangle] \cdot (id + \cdot f)
 \end{aligned}$$

Logo, $g = [\langle 0, 1 \rangle, \langle \pi_2, + \rangle]$. □

Anexo—Cálculo de Funções

COMPOSIÇÃO

$$\text{Natural-id} \quad f \cdot id = id \cdot f = f \quad (4)$$

$$\text{Associatividade} \quad (f \cdot g) \cdot h = f \cdot (g \cdot h) \quad (5)$$

PRODUTO

$$\text{Universal-}\times \quad k = \langle f, g \rangle \Leftrightarrow \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases} \quad (6)$$

$$\text{Cancelamento-}\times \quad \pi_1 \cdot \langle f, g \rangle = f \quad , \quad \pi_2 \cdot \langle f, g \rangle = g \quad (7)$$

$$\text{Reflexão-}\times \quad \langle \pi_1, \pi_2 \rangle = id_{A \times B} \quad (8)$$

$$\text{Fusão-}\times \quad \langle g, h \rangle \cdot f = \langle g \cdot f, h \cdot f \rangle \quad (9)$$

$$\text{Absorção-}\times \quad (i \times j) \cdot \langle g, h \rangle = \langle i \cdot g, j \cdot h \rangle \quad (10)$$

$$\text{Functor-}\times \quad (g \cdot h) \times (i \cdot j) = (g \times i) \cdot (h \times j) \quad (11)$$

$$\text{Functor-id-}\times \quad id_A \times id_B = id_{A \times B} \quad (12)$$

COPRODUTO

$$\text{Universal-+} \quad k = [f, g] \Leftrightarrow \begin{cases} k \cdot i_1 = f \\ k \cdot i_2 = g \end{cases} \quad (13)$$

$$\text{Cancelamento-+} \quad [g, h] \cdot i_1 = g \quad , \quad [g, h] \cdot i_2 = h \quad (14)$$

$$\text{Reflexão-+} \quad [i_1, i_2] = id_{A+B} \quad (15)$$

$$\text{Fusão-+} \quad f \cdot [g, h] = [f \cdot g, f \cdot h] \quad (16)$$

$$\text{Absorção-+} \quad [g, h] \cdot (i + j) = [g \cdot i, h \cdot j] \quad (17)$$

$$\text{Functor-+} \quad (g \cdot h) + (i \cdot j) = (g + i) \cdot (h + j) \quad (18)$$

$$\text{Functor-id-+} \quad id_A + id_B = id_{A+B} \quad (19)$$

EXPONENCIAÇÃO

$$\text{Universal} \quad k = \bar{f} \Leftrightarrow f = ap \cdot (k \times id) \quad (20)$$

$$\text{Cancelamento} \quad f = ap \cdot (\bar{f} \times id) \quad (21)$$

$$\text{Reflexão} \quad \overline{ap} = id_{B^A} \quad (22)$$

$$\text{Fusão} \quad \overline{g \cdot (f \times id)} = \bar{g} \cdot f \quad (23)$$

$$\text{Absorção} \quad f^A \cdot \bar{g} = \overline{f \cdot g} \quad (24)$$

$$\text{Functor} \quad (g \cdot h)^A = g^A \cdot h^A \quad (25)$$

$$\text{Functor-id} \quad id^A = id \quad (26)$$

INDUÇÃO

$$\text{Universal-cata} \quad k = \langle \alpha \rangle \Leftrightarrow k \cdot in = \alpha \cdot F k \quad (27)$$

$$\text{Cancelamento-cata} \quad \langle \alpha \rangle \cdot in = \alpha \cdot F \langle \alpha \rangle \quad (28)$$

$$\text{Reflexão-cata} \quad \langle in \rangle = id_{\mu F} \quad (29)$$

$$\text{Fusão-cata} \quad f \cdot \langle \alpha \rangle = \langle \beta \rangle \quad \text{if} \quad f \cdot \alpha = \beta \cdot F f \quad (30)$$

MISC.

Lei da troca

$$[\langle f, g \rangle, \langle h, k \rangle] = \langle [f, h], [g, k] \rangle \quad (31)$$

