CP/2122 - Resolução de exercícios nas aulas práticas

Índice: (páginas do PDF)

Cálculo de Programas (2021/22)

(Turnos TP2/TP5)

Fichas de apoio às aulas TP

(Aulas mais recentes primeiro)

Aula T [dúvidas] (13-Jan)

F11-Q5

5. Suponha um tipo indutivo $\mathsf{T}\ X$ cuja base é o bifunctor

$$\mathsf{B}\ (X, Y) = X + \mathsf{F}\ Y$$
$$\mathsf{B}\ (f, g) = f + \mathsf{F}\ g$$

onde F é um outro qualquer functor.

- Mostre que $\mathsf{T}\ X$ é um mónade em que

$$\mu = (\![[id\ , \mathsf{in} \cdot i_2]]\!)$$
$$u = \mathsf{in} \cdot i_1$$

onde $\mathsf{in} : \mathsf{B}\ (X, \mathsf{T}\ X) \to \mathsf{T}\ X$.

- Alguns mónades conhecidos, por exemplo LTree, resultam desta lei geral. Identifique F em cada caso.
- Para $\mathsf{F}\ Y = 1$ (e $\mathsf{F}\ f = id$) qual é o mónade que se obtém por esta regra? E no caso em que $\mathsf{F}\ Y = O \times Y^*$, onde o tipo O se considera fixo à partida?

Para evitar confusões com o functor F que descreve o padrão de recursividade dos catamorfismos etc, vamos renomear Facima para G, isto é

$$B(X, Y) \ = \ X \ + \ G\, Y$$

Para mostrar que T é monad, há que provar (61,62). Comecemos por (62):

- $\mu \cdot u \ = \ id$

- $\mu \cdot T\, u = id$

(62a) —------------------------------

$\mu \cdot u = id$

$\equiv$ { μ é $um\ cata$ }

$(\!|\, [id,\ in \cdot i2] \,|\!) \cdot in \cdot i_1 = id$

$\equiv$ { cancelamento-cata }

$[id,\ in \cdot i2] \cdot (id + G\,\mu) \cdot i_1 = id$

$\equiv$ { $absorção\ +$ }

$[id,\ in \cdot i2 \cdot G\,\mu] \cdot i_1 = id$

$\equiv$ { cancelamento-cata}

$id = id$

(62b) —--------------------------

$\mu \cdot T\,u = id$

$\equiv$ { μ é $um\ cata$ }

$(\!|\, [id,\ in \cdot i2] \,|\!) \cdot T\,u = id$

$\equiv$ { absorção-cata para $B(f, g) = f + G\,g$ }

$(\!|\, [id,\ in \cdot i2] \cdot (u + G\,id) \,|\!) = id$

$\equiv$ { $absorção\ +;\ G\,id = id$ }

$(\!|\, [u,\ in \cdot i2] \,|\!) = id$

$\equiv$ { definição de u}

$(\!|\, [in \cdot i_{:1},\ in \cdot i2] \,|\!) = id$

$\equiv$ { fusão + }

$$(\!| in \cdot [i_1, i_2] |\!) = id$$

$\equiv$ { reflexão +}

$$(\!| in |\!) = id$$

$\equiv$ { reflexão - cata }

$$true$$

(61) —---- TPC -----------------------

$$\mu \cdot \mu = \mu \cdot T\mu$$

$\equiv$ { $preencher$ }

$$\mu \cdot (\!| [id, in \cdot i2] |\!) = (\!| [id, in \cdot i2] |\!) \cdot T\mu$$

$\equiv$ { $preencher$ }

$$\mu \cdot (\!| [id, in \cdot i2] |\!) = (\!| [id, in \cdot i2] \cdot B(\mu, id) |\!)$$

$\Leftarrow$ { $preencher$ }

$$\mu \cdot [id, in \cdot i2] = [id, in \cdot i2] \cdot B(\mu, id) \cdot B(id, \mu)$$

$\equiv$ { $preencher$ }

$$\mu \cdot [id, in \cdot i2] = [id, in \cdot i2] \cdot (\mu + G\mu)$$

$\equiv$ { $preencher$ }

$$\mu \cdot in \cdot i2 = in \cdot i2 \cdot G\mu$$

$\equiv$ { $preencher$ }

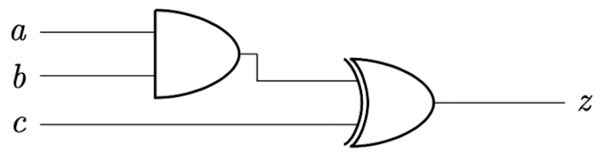
$$[id, in \cdot i2] \cdot (id + G\mu) \cdot i2 = in \cdot i2 \cdot G\mu$$

$\equiv$ { $preencher$ }

$$true$$

F01-Q7

5. Considere o circuito booleano



que calcula a função $f\ ((a,b),c) = (a \wedge b) \oplus c$, onde $\oplus$ é a operação "exclusive-or".

- Escreva uma definição dessa função $(\mathbb{B} \times \mathbb{B}) \times \mathbb{B} \xrightarrow{f} \mathbb{B}$ que não recorra às variáveis a, b ou c[1] e desenhe o respectivo diagrama.
- Qual é o tipo da função $g = \langle \pi_1, f \rangle$?

Função dada: $f\ ((a,b),c) \ = \ (a \ \wedge \ b) \ \oplus \ c$

Estratégia: arranjar forma de ambos os membros ficarem da forma $f\ ((a,b),c)$, por exemplo

$$f\ ((a,b),c) \ = \ \text{....}\ \alpha\ \text{...}\ ((a,b),c)$$

$$a \ \wedge \ b = (uncurry\ (\wedge))\ (a,b)$$
$$a \ \oplus \ b = (uncurry\ (\oplus))\ (a,b)$$

Então:

$$f\ ((a,b),c) \ = \ (a \wedge b) \ \oplus \ c$$

$\Leftrightarrow \{\ a \ \wedge \ b = (uncurry\ (\wedge))\ (a,b)\}$

$$f\ ((a,b),c) \ = \ (uncurry\ (\wedge))\ (a,b) \ \oplus \ c$$

$\Leftrightarrow \{\ d \ \oplus \ c \ = \ (uncurry\ (\oplus))\ (d,c) \ \}$

$$f\ ((a,b),c) \ = uncurry\ (\oplus)\ (uncurry\ (\wedge)\ (a,b)\ ,\ id\ c)$$

$\Leftrightarrow \{(f \ \times \ g)\ (x,y) \ = \ (f\ x,\ g\ y)$, lei (77) do formulário $\}$

$$f\ ((a,b),c)\ =\ uncurry\ (\oplus)\ ((uncurry\ (\wedge)\ \times\ id)\ ((a,b),c))$$

$\Leftrightarrow$ { Lei (72) }

$$\begin{tikzcd}
 & E & \\
D & D \times C \arrow[l, "\pi_1"'] \arrow[u, "\oplus"] \arrow[r, "\pi_2"] & C \\
 & (A \times B) \times C \arrow[ul, "\wedge \cdot \pi_1"] \arrow[u, "\wedge \times id"] \arrow[ur, "id \cdot \pi_2"'] &
\end{tikzcd}$$

$$f\ ((a,b),c)\ =\ (uncurry\ (\oplus)\ \cdot\ (uncurry\ (\wedge)\ \times\ id))\ ((a,b),c)$$

$\Leftrightarrow$ { Lei (71) }

$$f\ =\ uncurry\ (\oplus)\ \cdot\ (uncurry\ (\wedge)\ \times\ id)$$

---

[12] (última) Aula CP/TP5 (11-Jan)

F12-Q1

1. Recorde que o tipo $Maybe\ a = \mathsf{Just}\ a \mid \mathsf{Nothing}$ forma um mónade cuja operação de multiplicação pode ser captada pelo diagrama seguinte:

$$\begin{tikzcd}
Maybe\ (Maybe\ a) \arrow[d, "\mu"'] & (Maybe\ a) + 1 \arrow[l, "\mathsf{in}"'] \arrow[d, "id+!"] \\
Maybe\ a & (Maybe\ a) + 1 \arrow[l, "{[id,\ \mathbf{in} \cdot i_2]}"']
\end{tikzcd}
\qquad \mu \cdot \mathsf{in} = [id, \mathsf{in} \cdot i_2] \cdot (id + !)$$

onde $\mathsf{in} = [\mathsf{Just}, \mathsf{Nothing}]$. Derive deste diagrama a definição *pointwise* dessa função:

$$\mu\ (\mathsf{Just}\ a) = a$$
$$\mu\ \mathsf{Nothing} = \mathsf{Nothing}$$

Resolução:

$$\mu \cdot in = [id, in \cdot i_2] \cdot (id + !)$$

$\equiv$ { $definição\ de\ in$ }

$$\mu \cdot [Just, \underline{Nothing}] = [id, \underline{Nothing}] \cdot (id + !)$$

$\equiv$ { $fusão + à\ esquerda;\ absorção + à\ direita;\ natural - id$ }

$$[\mu \cdot Just, \mu \cdot \underline{Nothing}] = [id, \underline{Nothing}]$$

$\equiv$ { $Eq +$ }

$$\begin{cases} \mu \cdot Just = id \\ \mu \cdot \underline{Nothing} = \underline{Nothing} \end{cases}$$

$\equiv$ { $(71, 72)$ }

$$\begin{cases} \mu\,(Just\ a) = a \\ \mu\ Nothing = Nothing \end{cases}$$

F12-Q2

2. Repare que as projecções $A \xleftarrow{\pi_1} A \times B \xrightarrow{\pi_2} B$ são funções binárias e como tal podem ser "curried", $A^B \xleftarrow{\overline{\pi_1}} A \xrightarrow{\overline{\pi_2}} B^B$. Verifica-se que:

$$\overline{\pi_1} = \mathsf{const} \tag{F1}$$

$$\overline{\pi_2} = \underline{id} \tag{F2}$$

onde const $a = \underline{a}$, a função constante que dá a como resultado. Apresente justificações para os passos das provas respectivas que se seguem:

$$
\begin{array}{ll}
 & \overline{\pi_1} = \mathsf{const} \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & \mathsf{ap} \cdot (\mathsf{const} \times id) = \pi_1 \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & \mathsf{ap} \cdot (\mathsf{const} \times id)\ (a, b) = \pi_1\ (a, b) \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & \mathsf{ap}\ (\mathsf{const}\ a, b) = a \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & \mathsf{const}\ a\ b = a \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & \underline{a}\ b = a \\
\Box &
\end{array}
$$

$$
\begin{array}{ll}
 & \overline{\pi_2} = \underline{id} \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & \mathsf{ap} \cdot (\underline{id} \times id) = \pi_2 \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & \mathsf{ap}\ ((\underline{id} \times id)\ (a, b)) = b \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & \mathsf{ap}\ (id, b) = b \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
 & b = b \\
\Box &
\end{array}
$$

Resolução:

(a) $\overline{\pi_1} = const$

$\equiv$ (1) { (35) }
$\equiv$ (2) { (71) }
$\equiv$ (3) { (77, 79, 1) }
$\equiv$ (4) { (82) }
$\equiv$ (5) { *a notação const a é uma alternativa a* $\underline{a}$ }

(b) $\overline{\pi_2} = \underline{id}$

$\equiv$ (1) { (35) }
$\equiv$ (2) { (72,79) }
$\equiv$ (3) { (77, 1, 3) }
$\equiv$ (4) { (82) }

F12-Q3

3. Em Haskell, um mónade declara-se instanciando a classe $Monad$, onde se define a unidade u (que aí se designa por return) e uma operação $x \gg\!= f$, conhecida como aplicação monádica, ou *"binding"* de f a x, que é tal que

$$x \gg\!= f = (f \bullet id)\ x = (\mu \cdot \mathsf{T}\ f)\ x \tag{F3}$$

Mostre que:

$$\mu = (\gg\!= id) \tag{F4}$$
$$g \bullet f = (\gg\!= g) \cdot f \tag{F5}$$
$$x \gg\!= (f \bullet g) = (x \gg\!= g) \gg\!= f \tag{F6}$$

Resolução:

(F4)

$$\mu = (>>= \ id)$$

$\equiv \{ \ (71) \ \}$

$$\mu\, x = (>>= \ id)\ x$$

$\equiv \{$ *aplicação de secção de um operador curried* (>>=) $\}$

$$\mu\, x = x \ >>= \ id$$

$\equiv \{ \ (86) \ \}$

$$\mu\, x = (\mu \ \cdot \ T\ id)\ x$$

$\equiv \{ \ (44),\ (1) \ \}$

$$\mu\, x = \mu \ x$$

$\equiv \{ \ propriedade\ reflexiva\ da\ igualdade \ \}$

$$true$$

(F5)

$$g \bullet f = (>>= \ g) \ \cdot f$$

$\equiv \{ \ (71) \ \}$

$$(g \bullet f)\, x \;=\; ((>>= \;\; g) \;\cdot\, f)\, x$$

$\equiv$ { $(72),\ (65)$ }

$$(\mu \;\cdot\; T\, g \;\cdot\; f)\, x \;=\; (>>= \;\; g)\, (f\, x)$$

$\equiv$ { $secção\ (>>= \;\; g)$ }

$$(\mu \;\cdot\; T\, g \;\cdot\; f)\, x \;=\; (f\, x) \;\; >>= \;\; g$$

$\equiv$ { (86) }

$$(\mu \;\cdot\; T\, g \;\cdot\; f)\, x \;=\; (\mu \;\cdot T\, g)(f\, x)$$

$\equiv$ { (72) }

$$(\mu \;\cdot\; T\, g \;\cdot\; f)\, x \;=\; (\mu \;\cdot T\, g \cdot f)\, x$$

(F6)

$$x \;\; >>= \;\; (f \;\; \bullet\, g) \;=\; (x \;\; >>= \;\; g) \;\; >>= \;\; f$$

$\equiv$ { $(86) \;\; \times \;\; 2$ }

$$(\mu \;\cdot T(f \;\; \bullet\, g))\, x \;=\; (\mu \cdot T\, f)(x \;\; >>= \;\; g)$$

$\equiv$ { (86) à $direita$ }

$$(\mu \;\cdot T(f \;\; \bullet\, g))\, x \;=\; (\mu \cdot T\, f)((\mu \;\cdot T\, g)\, x)$$

$\equiv$ { (72) à $direita$; (71) em sentido inverso do habitual }

$$\mu \;\cdot T(f \;\; \bullet\, g) \;=\; \mu \cdot T\, f \,\cdot\, \mu \;\cdot T\, g$$

$\equiv$ { (65) }

$$\mu \;\cdot T(\mu \cdot \;\; T\, f \;\; \cdot g) \;=\; \mu \cdot T\, f \,\cdot\, \mu \;\cdot T\, g$$

$\equiv$ { (43) }

$$\mu \;\cdot T\, \mu \cdot \;\; T(T\, f) \;\; \cdot T\, g \;=\; \mu \cdot T\, f \,\cdot\, \mu \;\cdot T\, g$$

$\equiv$ { (64) }

$$\mu \;\cdot T\, \mu \cdot \;\; T(T\, f) \;\; \cdot T\, g \;=\; \mu \cdot \;\mu \;\cdot T(T\, f) \cdot T\, g$$

$\equiv$ { (61) }

$$\mu \cdot \mu \cdot T(T\,f) \cdot T\,g = \mu \cdot \mu \cdot T(T\,f) \cdot T\,g$$

$\equiv \{$ *propriedade reflexiva da igualdade* $\}$

$$true$$

F12-Q4

4. Sempre que um functor T é um mónade tem-se:

$$\mathsf{T}\,f = (u \cdot f) \bullet id$$

Definindo-se

$$\theta\ b = \mathsf{T}\ \langle \underline{b}, id \rangle$$

mostre que

$$\theta\ b\ x = \textbf{do}\ \{a \leftarrow x; \mathsf{return}\ (b, a)\} \tag{F7}$$

O que faz o operador θ? E qual a sua relação com o operador $lstr$ que consta da biblioteca Cp.hs? (**Sugestão**: use

$$(f \bullet g)\ a \quad = \quad \textbf{do}\ \{b \leftarrow g\ a; f\ b\} \tag{F8}$$

e outras leis que conhece do cálculo de mónades.)

Resolução: preparação

$$(T\,f)\,x = ((return \cdot f) \bullet id)\,x$$

$\equiv \{\ (F8)\ \}$

$$(T\,f)\,x = do\,\{\,b \leftarrow x;\ (return \cdot f)\,b\,\}$$

$\equiv \{$ *preencher* $\}$

$$(T\,f)\,x = do\,\{\,b \leftarrow x;\ return(f\ b)\,\}\ (**)$$

NB: Em listas, do { b <- x; return(f b) } coincide com [f b | b <- x]

$$\theta\, b\, x \;=\; T\,\langle \underline{b}\,,\, id\rangle\, \mathrm{x}$$

$\equiv \{\; (**)\; \}$

$$\theta\, b\, x \;=\; do\,\{\, a \;<\!-\; x;\; return(\,\langle \underline{b}\,,\, id\rangle\, a)\,\}$$

$\equiv \{\; (76)\; \}$

$$\theta\, b\, x \;=\; do\,\{\, a \;<\!-\; x;\; return((\underline{b}\, a\,,\, a))\,\}$$

$\equiv \{\; preencher\; \}$

$$\theta\, b\, x \;=\; do\,\{\, a \;<\!-\; x;\; return(b,\; a)\,\}$$

F12-Q5

5. Em Haskell, a instância para listas da função monádica sequence : $\mathsf{F}\ (\mathsf{T}\ X) \to \mathsf{T}\ (\mathsf{F}\ X)$ é o catamorfismo

$$\begin{array}{l} \mathsf{sequence} = (\!|\, g\, |\!)\ \mathbf{where} \\ \quad g = [\mathsf{return}, id] \cdot (\mathsf{nil} + \mathsf{cons}^\flat) \\ \quad f^\flat\ (x, y) = \mathbf{do}\ \{ a \leftarrow x; b \leftarrow y; \mathsf{return}\ (f\ (a, b)) \} \end{array}$$

tal como se mostra neste diagrama:

$$\begin{tikzcd}
(\mathsf{T}\ X)^* \arrow[d, "\mathsf{sequence}"'] & 1 + \mathsf{T}\ X \times (\mathsf{T}\ X)^* \arrow[l, "\mathsf{in}"'] \arrow[d, "id + id \times \mathsf{sequence}"] \\
\mathsf{T}\ (X^*) & 1 + \mathsf{T}\ X \times \mathsf{T}\ (X^*) \arrow[l, dotted, "g"'] \arrow[d, "\mathsf{nil} + \mathsf{cons}^\flat"] \\
 & X^* + \mathsf{T}\ (X^*) \arrow[ul, dotted, "{[\mathsf{return}, id]}"]
\end{tikzcd}$$

Partindo da propriedade universal-cata, derive uma versão de sequence em Haskell com variáveis que não recorra à composição de funções.

Resolução:

$$sequence \;=\; (\!|\, [return, id]\,.\,(nil \;+\; cons')\,|\!)$$

$\equiv \{\; preencher\; \}$

$$sequence = (\!| [return \cdot nil, cons'] |\!)$$

$\equiv \{ \ universal - cata \}$

$$sequence \cdot in = [return \cdot nil, cons'] \cdot (id + id \times sequence)$$

$\equiv \{ \ preencher \}$

$$[sequence \cdot nil, \ sequence \cdot cons] = [return \cdot nil, cons' \cdot (id \times sequence)]$$

$\equiv \{ \ Eq -+ \}$

$$\begin{cases} sequence \cdot nil = return \cdot nil \\ sequence \cdot cons = cons' \cdot (id \times sequence) \end{cases}$$

$\equiv \{ \ (71, 72) \}$

$$\begin{cases} sequence\ [] = return\ [] \\ sequence\ (cons(h, t)) = cons'\ (h, \ sequence\ t) \end{cases}$$

$\equiv \{ \ preencher \}$

$$\begin{cases} sequence\ [] = return\ [] \\ sequence\ (h{:}\ t) = do \{ \ a <- \ h\ ; \ b <- \ sequence\ t; \ return(a{:}\ b) \} \end{cases}$$

---

[11] Aula CP/TP5 (4-Jan)

F08-Q1

1. Considere a função

$$\mathsf{mirror}\ (Leaf\ \ a) = Leaf\ \ a$$
$$\mathsf{mirror}\ (Fork\ (x, y)) = Fork\ (\mathsf{mirror}\ y, \mathsf{mirror}\ x)$$

que "espelha" árvores binárias do tipo LTree (ver fichas anteriores). Comece por mostrar que

$$\mathsf{mirror} = (\!|\, \mathsf{in} \cdot (id + \mathsf{swap})\, |\!) \qquad \text{(F1)}$$

desenhando o digrama que representa este catamorfismo.

Tal como swap, mirror é um isomorfismo de árvores pois é a sua própria inversa:

$$\mathsf{mirror} \cdot \mathsf{mirror} = id \qquad \text{(F2)}$$

Complete a seguinte demonstração de (F2):

$$\begin{array}{cl} & \mathsf{mirror} \cdot \mathsf{mirror} = id \\ \equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\ & \mathsf{mirror} \cdot (\!|\, \mathsf{in} \cdot (id + \mathsf{swap})\, |\!) = (\!|\, \mathsf{in}\, |\!) \\ \Leftarrow & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\ & \mathsf{mirror} \cdot \ldots = \ldots \\ \ldots & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\ & (etc) \end{array}$$

Resolução:

$$mirror \cdot mirror = id$$

$\equiv \{$ $lei\ (47)$ à $direita;\ mirror = (\!|\, in \cdot (id + swap)\, |\!)$ $\}$

$$mirror \cdot (\!|\, in \cdot (id + swap)\, |\!) = (\!|\, in\, |\!)$$

$\Leftarrow \{\ lei\ de\ fusão - cata\ \}$

$$mirror \cdot in \cdot (id + swap) = in \cdot F\, mirror$$

$\equiv \{\ F\, f = B(id, f)\ para\ B(g, f) = g + f \times f\ \}$

$$mirror \cdot in \cdot (id + swap) = in \cdot (id + mirror^2)$$

$\equiv \{\ mirror = (\!|\, in \cdot (id + swap)\, |\!)$; Cancelamento-cata - 46$\}$

$$in \cdot (id + swap) \cdot (id + mirror^2) \cdot (id + swap) = in \cdot (id + mirror^2)$$

$\equiv \{\ Functor\ \ + \}$

$$in\ \cdot (id + swap\ \cdot\ mirror^2\ \cdot swap)\ =\ in\ \cdot (id\ +\ mirror^2)$$

$\equiv \{\ grátis\ de\ swap: swap\ \cdot (f\ \times\ g)\ =\ (g\ \times\ f) \cdot swap\ \}$

$$in\ \cdot (id + swap\ \cdot swap \cdot\ mirror^2)\ =\ in\ \cdot (id\ +\ mirror^2)$$

$\equiv \{\ swap\ \cdot\ swap\ =\ id \}$

$$in\ \cdot (id +\ mirror^2)\ =\ in\ \cdot (id\ +\ mirror^2)$$

$\equiv \{\ trivial \}$

$$true$$

F08-Q5

5. Um *bifunctor* B é um functor **binário**

$$\begin{tikzcd} A \arrow[d, "f"'] & C \arrow[d, "g"'] & \mathsf{B}\,(A, C) \arrow[d, "\mathsf{B}\,(f,g)"] \\ D & E & \mathsf{B}\,(D, E) \end{tikzcd} \quad \text{tal que:} \quad \left\{ \begin{array}{l} \mathsf{B}\,(id, id) = id \\ \mathsf{B}\,(f \cdot g, h \cdot k) = \mathsf{B}\,(f, h) \cdot \mathsf{B}\,(g, k) \end{array} \right. \qquad \text{(F4)}$$

Mostre que $\mathsf{B}\,(X, Y) = X \times Y$, $\mathsf{B}\,(X, Y) = X + Y$ e $\mathsf{B}\,(X, Y) = X + Y \times Y$ são bifunctores.

$B(X, Y)\ =\ X\ \times\ Y,\ B(f, g)\ =\ f\ \ \times\ \ g$

$$B(id, id)\ =\ id$$

$\equiv \{\ B(X, Y)\ =\ X\ \times\ Y \}$

$$id\ \times id\ =\ id$$

$\equiv \{\ Functor\ -\ id\ -\times\ (15) \}$

$$true$$

2ª lei:

$$B(f\ \cdot\ g,\ h \cdot\ k)\ =\ B(f, h)\ \cdot B(g, k)$$

$\equiv \{\ B(X,Y) \ = \ X \ \times \ Y \ \}$

$$(f \ \cdot \ g) \times \ (h \ \cdot \ k) \ = (\ f \ \times \ h) \ \cdot (g \ \times \ k)$$

$\equiv \{\ Functor \ -\times \ (14)\ \}$

$$true$$

2º caso: $B(X,Y) \ = \ X \ + \ Y \ \times \ Y$

$\equiv \{\ B(id, id) \ = \ id\ \}$

$$id \ + \ id \ \times id \ = \ id$$

$\equiv \{\ Functor \ - \ id \ -\times \ (15)\ \}$

$$id \ + \ id \ = \ id$$

$\equiv \{\ Functor \ - \ id \ -+ \ (26)\ \}$

$$true$$

2º caso: $B(X,Y) \ = \ X \ + \ Y \ \times \ Y, \ isto$ é, B é $o\ bifunctor\ das\ LTrees$

$$B(f \ \cdot \ g, \ h \ \cdot \ k) \ = \ B(f,h) \ \cdot B(g,k)$$

$\equiv \{\ B(X,Y) \ = \ X \ + \ Y \ \times \ Y\}$

$$(f \ \cdot \ g) \ + \ ((\ h \ \cdot \ k) \times (h \ \cdot \ k)) \ = \ (f \ + \ (h \times h)) \cdot (g \ + (k \times k))$$

$\equiv \{\ Functor \ -\times \ \}$

$$(f \ \cdot \ g) \ + \ ((\ h \times \ h) \ \cdot (k \times \ k)) \ = \ (f \ + \ (h \times h)) \cdot (g \ + (k \times k))$$

$\equiv \{$ Functor-+ à direita$\}$

$$(f \ \cdot \ g) \ + \ ((\ h \times \ h) \ \cdot (k \times \ k)) \ = \ (f \ \cdot \ g) \ + \ ((\ h \times \ h) \ \cdot (k \times \ k))$$

$\equiv \{$ trivial $\}$

$$true$$

F10-Q7(b)

7. Nas aulas teóricas viu-se que, sempre que um ciclo-*while* termina, ele pode ser definido por

$$\textbf{while}\ p\ f\ g = \textbf{tailr}\ ((g + f) \cdot (\neg \cdot p)?) \qquad \text{(F3)}$$

recorrendo ao combinador de *"tail recursion"* **tailr** $f = [\![\nabla, f]\!]$, que é um hilomorfismo de base $\mathsf{B}\,(X, Y) = X + Y$, para $\nabla = [id\,, id]$.

(a) Derive a definição *pointwise* de **while** $p\ f\ g$, sabendo que qualquer $h = [\![f, g]\!]$ é tal que $h = f \cdot \mathsf{F}\,h \cdot g$.

(b) Complete a demonstração da lei de fusão de **tailr**[2]

$$(\textbf{tailr}\ g) \cdot f = \textbf{tailr}\ h \Leftarrow (id + f) \cdot h = g \cdot f$$

Sabemos que $tailr\ f = [\![\nabla, f]\!] = (\!|\nabla|\!) \cdot [\!(f)\!]$. Então:

$$(tailr\ g) \cdot f = tailr\ h$$

$\equiv \{\ definição\ de\ tailr\ \}$

$$(\!|\nabla|\!) \cdot [\!(g)\!] \cdot f = (\!|\nabla|\!) \cdot [\!(h)\!]$$

$\Leftarrow \{\ Leibniz\ (5)\ \}$

$$[\!(g)\!] \cdot f = [\!(h)\!]$$

$\Leftarrow \{\ Fusão - ana (57)\ \}$

$$g \cdot f = F\,f \cdot h$$

$\equiv \{\ B(X,Y) = X + Y;\ F\,f = B(id, f)\ \}$

$$g \cdot f = (id + f) \cdot h$$

F11-Q1

1. O facto de length: $A^* \to \mathbb{N}_0$ poder ser definida tanto como *catamorfismo* de listas como *anamorfismo* de naturais (que foi assunto de uma questão de uma ficha anterior) pode generalizar-se da forma seguinte: sejam dados dois tipos indutivos

$$\begin{tikzcd} \mathsf{T}_1 \arrow[r, bend left, "\mathsf{out}_1"] \arrow[r, phantom, "\cong"] & \mathsf{F}\ \mathsf{T}_1 \arrow[l, bend left, "\mathsf{in}_1"] \end{tikzcd} \qquad \begin{tikzcd} \mathsf{T}_2 \arrow[r, bend left, "\mathsf{out}_2"] \arrow[r, phantom, "\cong"] & \mathsf{G}\ \mathsf{T}_2 \arrow[l, bend left, "\mathsf{in}_2"] \end{tikzcd}$$

e $\alpha : \mathsf{F}\ X \to \mathsf{G}\ X$, isto é, α satisfaz a propriedade *grátis*

$$\mathsf{G}\ f \cdot \alpha = \alpha \cdot \mathsf{F}\ f \tag{F1}$$

Então $(\!|\, \mathsf{in}_2 \cdot \alpha \,|\!) = [\!(\alpha \cdot \mathsf{out}_1)\!]$, como se mostra a seguir (complete as justificações):

$$\begin{array}{ll}
& k = (\!|\, \mathsf{in}_2 \cdot \alpha \,|\!) \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& k \cdot \mathsf{in}_1 = \mathsf{in}_2 \cdot \alpha \cdot \mathsf{F}\ k \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mathsf{out}_2 \cdot k = \mathsf{G}\ k \cdot \alpha \cdot \mathsf{out}_1 \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& k = [\!(\alpha \cdot \mathsf{out}_1)\!] \\
\Box &
\end{array}$$

Identifique T_1, T_2 e α para o caso de $k = \mathsf{length}$.

Resolução:

$$k = (\!|\, in_2 \cdot \alpha \,|\!)$$

$\equiv \{ \ Universal - cata\ (45) \}$

$$k \cdot in_1 = in_2 \cdot \alpha \cdot F\,k$$

$\equiv \{ \ (33);\ (34); \}$

$$out_2 \cdot k = \alpha \cdot F\,k \cdot out_1$$

$\equiv \{ \ \alpha \cdot F\,f = G\,f \cdot \alpha; \}$

$$out_2 \cdot k = G\,k \cdot \alpha \cdot out_1$$

$\equiv \{ \ Universal - ana\ (54) \}$

$$k = [\![\alpha \cdot out_1]\!]$$

Caso particular $k = length$:

$$T1 = A*$$
$$T2 = N_0$$
$$F\,f = id + id \times f$$
$$G\,f = id + f$$
$$\alpha = ?$$

$length: A* \rightarrow No,\ logo\ T1 = A*\ e\ T2 = No$

$$in_1 = [nil, cons]$$
$$in_2 = [zero, succ]$$

$\alpha: (1 + A \times B) \rightarrow (1 + B)$
$\alpha = id + \pi2$

$length = (\![[zero, succ] \cdot (id + \pi2)]\!) = (\![[zero, succ \cdot \pi_2]]\!)$

$length = [\![(id + \pi_2) \cdot out_1]\!]$

F11-Q3

3. Um mónade é um functor T equipado com duas funções μ e u,

$$A \xrightarrow{u} \mathsf{T}\ A \xleftarrow{\mu} \mathsf{T}\ (\mathsf{T}\ A)$$

que satisfazem (para além das naturais, ie. "grátis") as propriedades $\mu \cdot u = id = \mu \cdot \mathsf{T}\ u$ e $\mu \cdot \mu = \mu \cdot \mathsf{T}\ \mu$ — identifique-as no formulário — com base nas quais se pode definir a *composição monádica*:

$$f \bullet g = \mu \cdot \mathsf{T}\ f \cdot g.$$

(Identifique-a também no formulário.) Demonstre os factos seguintes:

$$\mu = id \bullet id \quad \text{(F3)}$$
$$f \bullet u = f \quad \wedge \quad f = u \bullet f \quad \text{(F4)}$$
$$(f \cdot g) \bullet h = f \bullet (\mathsf{T}\ g \cdot h) \quad \text{(F5)}$$
$$\mathsf{T}\ f = (u \cdot f) \bullet id \quad \text{(F6)}$$

(F3)

$\mu = id \bullet id$

$\equiv$ { Composição monádica - 65}

$\mu = \mu \cdot T\,id \cdot id$

$\equiv \{\ Functor - id - F(44);\ Natural - id\ (1)\ \}$

$\mu = \mu$

$\equiv \{\ trivial\ \}$

$true$

(F5)

$(f \cdot g) \bullet h = f \bullet (T\,g \cdot h)$

$\equiv \{\ (65)\ duas\ vezes\ \}$

$\mu \cdot T(f \cdot g) \cdot h = \mu \cdot T\,f \cdot T\,g \cdot h$

$\equiv \{\ Functor - F\ (43)\ \}$

$$\mu \cdot T\,f \cdot T\,g \cdot h = \mu \cdot T\,f \cdot T\,g \cdot h$$

$\equiv$ { $trivial$ }

$$true$$

(F6)

$$T\,f = (u \cdot f) \bullet id$$

$\equiv$ { (65) }

$$T\,f = \mu \cdot T\,(u \cdot f)$$

$\equiv$ { (43) }

$$T\,f = \mu \cdot T\,u \cdot T\,f$$

$\equiv$ { Unidade - (62) }

$$T\,f = id \cdot T\,f$$

$\equiv$ { $Natural - id$ }

$$T\,f = T\,f$$

$\equiv$ { trivial }

$$true$$

F11-Q4

4. A função $discollect : (A \times B^*)^* \to (A \times B)^*$ que apareceu (sem ser definida) numa questão das primeiras fichas não é mais do que

$$discollect = lstr \bullet id \tag{F7}$$

— onde $lstr\ (a, x) = [(a, b) \mid b \leftarrow x]$ — no mónade das listas, $\mathsf{T}\ A = A^*$,

$$A \xrightarrow{\ \mathsf{singl}\ } A^* \xleftarrow{\ \mathsf{concat}\ } (A^*)^*$$

onde $u = \mathsf{singl}$ e $\mu = \mathsf{concat} = (\![\mathsf{nil}\ , \mathsf{conc}]\!)$. Recordando a lei de absorção-cata (para listas), derive uma definição recursiva para *discollect* que não use nenhum dos combinadores 'point-free' estudados nesta disciplina.

Resolução:

$$discollect = lstr \bullet id$$

$\equiv$ { (65) }

$$discollect = \mu \cdot T\ lstr \cdot id$$

$\equiv$ { Def - μ para listas }

$$discollect = concat \cdot T\ lstr$$

$\equiv$ { $Def - concat\ como\ cata$ }

$$discollect = (\![nil, conc]\!) \cdot T\ lstr$$

$\equiv$ { Absorção-cata (51)
$para\ B(f, g) = id + f \times g;\ Natural - id(1);\ Absorção -+ (22)$ }

$$discollect = (\![nil, conc \cdot (lstr \times id)]\!)$$

$\equiv$ { $Universal - cata\ (45);\ Definição\ de\ in = [nil, cons];$ Fusão-+ à esquerda(20); }

$$[discollect \cdot nil, discollect \cdot cons] = [nil, conc \cdot (lstr \times id)] \cdot F\ discollect$$

$\equiv$ { F discollect = id + id x discollect; Absorção-+(22) ;Eq-+(27) }

$$discollect \cdot nil = nil$$

$$discollect \cdot cons = conc \cdot (lstr \times id) \cdot (id \times discollect)$$

$\equiv \{ \ (71, 72) \ ; \ nil \ x = []; \ cons\ (a, b) = a\colon b; \ Functor - x(14); \ Def - x\ (77); \}$

$$discollect\ [] = []$$

$$discollect(h\colon t) = conc\ \ (lstr\ h, \ discollect\ t)$$

$\equiv \{ \ mud.\ de\ var.\ h := (a, x)\ pois\ a\ lista\ de\ entrada\ \text{é}\ de\ pares; \ conc\ (a, b) = a \ {+}{+}\ b;$
$\}$

$$discollect\ [] = []$$

$$discollect((a, x)\colon t) = lstr(a, x) \ {+}{+}\ discollect\ t$$

$\equiv \{ \ defini\text{ção}\ de\ lstr \}$

$$discollect\ [] = []$$

$$discollect\ ((a, x)\colon t) = [(a, \ b) \mid b \leftarrow x\ ] \ {+}{+}\ discollect\ t$$

NB: [f a | a <- x] = do { a <-x ; return(f a) }

---

[10] Aula CP/TP2 (14-Dez) - tema: anas + hilos

F09-Q6 (anamorfismos)

6. Um *anamorfismo* é um *"catamorfismo ao contrário"*, isto é, uma função $k : A \to \mathsf{T}$ tal que

$$k = \mathsf{in} \cdot \mathsf{F}\ k \cdot g \tag{F7}$$

escrevendo-se $k = [\!(g)\!]$. Mostre que o anamorfismo de listas

$$k = [\!((id + \langle f, id\rangle) \cdot \mathsf{out}_{\mathbb{N}_0})\!] \tag{F8}$$

descrito pelo diagrama

$$\begin{tikzcd}
\mathbb{N}_0{}^* & & 1 + \mathbb{N}_0 \times \mathbb{N}_0{}^* \arrow[ll, "\mathsf{in}"'] \\
\mathbb{N}_0 \arrow[u, "k"] \arrow[r, "\mathsf{out}_{\mathbb{N}_0}"'] & 1 + \mathbb{N}_0 \arrow[r, "id+\langle f, id\rangle"'] & 1 + \mathbb{N}_0 \times \mathbb{N}_0 \arrow[u, "id+id\times k"']
\end{tikzcd}$$

é a função

$$\begin{array}{l} k\ 0 = [\,] \\ k\ (n+1) = (2\ n + 1) : k\ n \end{array}$$

para $f\ n = 2\ n + 1$. (Que faz esta função?)

Resolução:

Universal-ana	$k = [\!(g)\!] \quad \Leftrightarrow \quad \mathsf{out} \cdot k = (\mathsf{F}\ k) \cdot g$	(54)
Cancelamento-ana	$\mathsf{out} \cdot [\!(g)\!] = \mathsf{F}\ [\!(g)\!] \cdot g$	(55)
Reflexão-ana	$[\!(\mathsf{out})\!] = id_{\mathsf{T}}$	(56)
Fusão-ana	$[\!(g)\!] \cdot f = [\!(h)\!] \quad \Leftarrow \quad g \cdot f = (\mathsf{F}\ f) \cdot h$	(57)
Base-ana	$\mathsf{F}\ f \quad = \quad \mathsf{B}\ (id, f)$	(58)
Def-map-ana	$\mathsf{T}\ f \quad = \quad [\!(\mathsf{B}(f, id) \cdot \mathsf{out})\!]$	(59)
Absorção-ana	$\mathsf{T}\ f \cdot [\!(g)\!] \quad = \quad [\!(\mathsf{B}(f, id) \cdot g)\!]$	(60)

$$k = [\!((id + \langle f, id\rangle) \cdot out_{\mathbb{N}})\!]$$

$\equiv \{\ (54)\ \}$

$$out \cdot k = (id + id \times k) \cdot (id + \langle f, id\rangle) \cdot out_{\mathbb{N}}$$

$\equiv \{\ iomorfismos\ in/out\ (33, 34)\ \}$

$$k \cdot in_{\mathbb{N}} = in \cdot (id + id \times k) \cdot (id + \langle f, id\rangle)$$

$\equiv \{\ definições\ de\ in\ (naturais\ e\ listas)\ \}$

$$k \cdot [zero, succ] = [nil, cons] \cdot (id + id \times k) \cdot (id + \langle f, id \rangle)$$

$\equiv \{\ fusão\ +\ \text{à}\ esquerda;\ absorção\ +\ \text{à}\ direita;\ natural - id\ \}$

$$[k \cdot zero, k \cdot succ] = [nil, cons \cdot (id \times k)] \cdot (id + \langle f, id \rangle)$$

$\equiv \{\ absorção\ +\ \}$

$$[k \cdot zero, k \cdot succ] = [nil, cons \cdot (id \times k) \cdot \langle f, id \rangle]$$

$\equiv \{\ Eq\ +\ \}$

$$\begin{aligned} k \cdot zero &= nil \\ k \cdot succ &= cons \cdot (id \times k) \cdot \langle f, id \rangle \end{aligned}$$

$\equiv \{\ absorção\ \times\ \}$

$$\begin{aligned} k \cdot zero &= nil \\ k \cdot succ &= cons \cdot \langle f, k \rangle \end{aligned}$$

$\equiv \{\ (71, 72, 76)\ \}$

$$\begin{aligned} k\ (zero\ x) &= nil\ x \\ k(succ\ x) &= cons(f\ x,\ k\ x) \end{aligned}$$

$\equiv \{\ definições\ das\ funções\ zero,\ nil, succ, cons\ \}$

$$\begin{aligned} k\ 0 &= [] \\ k(x + 1) &= f\ x : k\ x \end{aligned}$$

$\equiv \{\ assumindo\ f\ x = 2\ x + 1\ \}$

$$\begin{aligned} k\ 0 &= [] \\ k\ (x + 1) &= (2x + 1) : k\ x \end{aligned}$$

F10-Q2

1. O formulário desta disciplina apresenta duas definições alternativas para o functor $\mathsf{T}\,f$ de um tipo indutivo, uma como *catamorfismo* e outra como *anamorfismo*. Identifique-as e acrescente justificações à seguinte prova de que essas definições são equivalentes:

$$
\begin{array}{ll}
& \mathsf{T}\,f = (\!|\, \mathsf{in} \cdot \mathsf{B}\,(f, id) \,|\!) \\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \} \\
& \mathsf{T}\,f \cdot \mathsf{in} = \mathsf{in} \cdot \mathsf{B}\,(f, id) \cdot \mathsf{F}\,(\mathsf{T}\,f) \\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \} \\
& \mathsf{T}\,f \cdot \mathsf{in} = \mathsf{in} \cdot \mathsf{B}\,(id, \mathsf{T}\,f) \cdot \mathsf{B}\,(f, id) \\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \} \\
& \mathsf{out} \cdot \mathsf{T}\,f = \mathsf{F}\,(\mathsf{T}\,f) \cdot \mathsf{B}\,(f, id) \cdot \mathsf{out} \\
\equiv & \quad \{ \quad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad \} \\
& \mathsf{T}\,f = [\!(\, \mathsf{B}\,(f, id) \cdot \mathsf{out} \,)\!] \\
\Box &
\end{array}
$$

Resolução:

$$T\,f \;=\; (\!|\, in \;\cdot\; B(f, id) \,|\!)$$

$\equiv \{\; universal - cata \,\}$

$$T\,f \;\cdot\; in = \; in \;\cdot\; B(f, id) \;\cdot\; F(T\,f)$$

$\equiv \{\; (49) \,\}$

$$T\,f \;\cdot\; in = \; in \;\cdot\; B(f, id) \;\cdot\; B(id, T\,f)$$

$\equiv \{\; (43)\; para\; bifunctores \,\}$

$$T\,f \;\cdot\; in = \; in \;\cdot\; B(f \;\cdot\; id, id \;\cdot\; T\,f)$$

$\equiv \{\; natural - id \,\}$

$$T\,f \;\cdot\; in = \; in \;\cdot\; B(id \;\cdot\; f, T\,f \;\cdot\; id)$$

$\equiv \{\; (43)\; para\; bifunctores \,\}$

$$T\,f \;\cdot\; in = in \;\cdot\; B(id, T\,f) \;\cdot\; B(f, id)$$

$\equiv \{\; (33, 34, 49) \,\}$

$$out \cdot T\, f \;=\; F\,(T\, f) \;\cdot\; B(f, id) \;\cdot\; out$$

$\equiv \{\; universal - ana \; para \; k := \; T\, f \; e \; g := \; B(f, id) \;\cdot\; out \,\}$

$$T\, f \;=\; [\![\, B(f, id) \;\cdot\; out \,]\!]$$

Universal-ana	$k = [\!(g)\!] \;\Leftrightarrow\; \mathsf{out} \cdot k = (\mathsf{F}\, k) \cdot g$	(54)
Cancelamento-ana	$\mathsf{out} \cdot [\!(g)\!] = \mathsf{F}\, [\!(g)\!] \cdot g$	(55)
Reflexão-ana	$[\!(\mathsf{out})\!] = id_{\mathsf{T}}$	(56)
Fusão-ana	$[\!(g)\!] \cdot f = [\!(h)\!] \;\Leftarrow\; g \cdot f = (\mathsf{F}\, f) \cdot h$	(57)
Base-ana	$\mathsf{F}\, f \;=\; \mathsf{B}\,(id, f)$	(58)
Def-map-ana	$\mathsf{T}\, f \;=\; [\!(\mathsf{B}(f, id) \cdot \mathsf{out})\!]$	(59)
Absorção-ana	$\mathsf{T}\, f \cdot [\!(g)\!] \;=\; [\!(\mathsf{B}(f, id) \cdot g)\!]$	(60)

F10-Q3

2. Mostre que o catamorfismo de listas $\mathsf{length} = (\![\mathsf{zero}\,, \mathsf{succ} \cdot \pi_2]\!)$ é a mesma função que o *anamorfismo* de naturais $[\!((id + \pi_2) \cdot \mathsf{out}_{\mathsf{List}})\!]$.

Resolução:

$$length \;=\; (\!|\, [\, zero,\; succ \,.\, \pi 2 \,] \,|\!)$$

$\equiv \{\; universal - cata \,\}$

$$length \cdot in_{\,List} \;=\; [\, zero,\; succ \,.\, \pi 2 \,] \;\cdot\; (id \;+\; id \;\times\; length)$$

$\equiv \{\; absorção \; + \}$

$$length \cdot in_{\,List} \;=\; [\, zero,\; succ \,.\, \pi 2 \;\;\cdot\; (id \;\times\; length)]$$

$\equiv \{\; natural \; \pi 2 \;\}$

$$length \cdot in_{\,List} \;=\; [zero, succ \;\cdot\; length \;\cdot\; \pi 2]$$

$\equiv \{\; in\, ;\; absorção \; + \}$

$$length \cdot in_{List} = in_{Nat} \cdot (id + length \cdot \pi2)$$

$\equiv \{ \ functor + \}$

$$length \cdot in_{List} = in_{Nat} (id + length) \cdot (id + \pi2) \cdot$$

$\equiv \{ \ (33, 34) \}$

$$outN \cdot length = (id + length) \cdot (id + \pi2) \cdot out_{List}$$

$\equiv \{ \ universal - ana \}$

$$length = [\!((id + \pi2) \cdot out_{List})\!]$$

F10-Q6

6. O algorítmo "bubble-sort" é o ciclo-for

$$\begin{array}{l} bSort\ xs = \mathsf{for\ bubble}\ xs\ (\mathsf{length}\ xs)\ \mathbf{where} \\ \quad \mathsf{bubble}\ (x : y : xs) \\ \qquad \mid x > y = y : \mathsf{bubble}\ (x : xs) \\ \qquad \mid \mathsf{otherwise} = x : \mathsf{bubble}\ (y : xs) \\ \quad \mathsf{bubble}\ x = x \end{array}$$

cujo corpo de ciclo é um hilomorfismo bubble $= [\![conquer, divide]\!]$. Identifique os genes $divide$ e $conquer$ desse hilomorfismo. **Sugestão**: siga a heurística que foi usada nas aulas teóricas para fazer o mesmo para a função de Fibonacci.[1]

```
bubble (x:y:xs)
        | x > y = y : bubble (x:xs)
        | otherwise = x : bubble (y:xs)
bubble x = x
```

Passo 1: retirar chamadas recursivas e renomear para divide

```
divide (a:b:as)
        | a > b = b  … (a:as)
        | otherwise = a … (b:as)
divide as = as
```

Passo 2: usar injecções para compatibilizar tipos de saída.

```
divide (a:b:as)
    | a > b = i2(b,(a:as))
    | otherwise = i2(a,(b:xs))
divide as = i1 as
```

Passo 3: identificar functor F

$$A* \rightarrow A* + A \times A*$$
$$F\,Y = A* + A \times Y$$
$$B(X, Y) = A* + X \times Y$$
$$F\,f = B(id, f) = id + id \times f$$

Passo 4: definir conquer

A* <- A* + A >< A*

$conquer = [g1, g2]\ \ para\ g1 = id;\ g2(a, as) = a{:}\,as$

Em suma:

- $bubble = [\![conquer, divide]\!]$
- $conquer = [id, cons]$

```
divide (a:b:as)
    | a > b = i2(b,(a:as))
    | otherwise = i2(a,(b:xs))
divide as = i1 as
```

F10-Q7

7. Nas aulas teóricas viu-se que, sempre que um ciclo-*while* termina, ele pode ser definido por

$$\mathbf{while}\ p\ f\ g = \mathbf{tailr}\ ((g+f)\cdot(\neg\cdot p)?) \qquad \text{(F3)}$$

recorrendo ao combinador de *"tail recursion"* $\mathbf{tailr}\ f = [\![\nabla, f]\!]$, que é um hilomorfismo de base $\mathsf{B}\ (X, Y) = X + Y$, para $\nabla = [id\ , id]$.

(a) Derive a definição *pointwise* de $\mathbf{while}\ p\ f\ g$, sabendo que qualquer $h = [\![f, g]\!]$ é tal que $h = f \cdot \mathsf{F}\ h \cdot g$.

(b) Complete a demonstração da lei de fusão de **tailr**[2]

$$(\mathbf{tailr}\ g)\cdot f = \mathbf{tailr}\ h \Leftarrow (id + f)\cdot h = g\cdot f$$

Resolução: (a) Pelo enunciado

- $tailr\ f = [\![\nabla, f]\!] = (\!|\nabla|\!)\cdot[\!(f)\!]$
- $B(X,Y) = X + Y$
- $B(f,g) = f + g$
- $F\ f = B(id, f) = id + f$
- $\nabla = [id, id]$.

Mais ainda, se

$h = [\![f, g]\!]$

então é válida a igualdade $h = f \cdot F\,h \cdot g$.

—-------------

$$while\ p\ f\ g = tailr\ ((g + f)\cdot(\neg p)?)$$

$\equiv \{\ tailr\ f = [\![\nabla, f]\!]\ \}$

$$while\ p\ f\ g = [\![\nabla, (g + f)\cdot(\neg p)?]\!]$$

$\equiv \{\ preencher\ \}$

$$while\ p\ f\ g = \nabla\cdot F\,(while\ p\ f\ g)\cdot(g + f)\cdot(\neg p)?$$

$\equiv \{\ preencher\ \}$

$$while\ p\ f\ g = [id, id]\cdot(id + while\ p\ f\ g)\cdot(g + f)\cdot(\neg p)?$$

$\equiv \{\ preencher\ \}$

$$while\ p\ f\ g\ = [id,\ while\ p\ f\ g] \cdot (g + f) \cdot (\neg p)?$$

$\equiv \{\ preencher\ \}$

$$while\ p\ f\ g\ = [g,\ ((while\ p\ f\ g) \cdot f)] \cdot (\neg p)?$$

$\equiv \{\ (30)\ \}$

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

$\equiv \{\ preencher\ \}$

$$(while\ p\ f\ g)\ x\ = (\neg p \rightarrow g,\ (while\ p\ f\ g) \cdot f)\text{x}$$

$\equiv \{\ (72),\ (78)\ \}$

$$(while\ p\ f\ g)\ x\ = if\ p\ x\ then\ while\ p\ f\ g(f\ x)\ else\ g\ x$$

(b) ver uma aula mais à frente

---

[09] Aula CP/TP5 (07-Dez) - tema: absorção-cata

F09-Q1

1. O diagrama genérico de um catamorfismo de gene g sobre o *tipo paramétrico* $\mathsf{T}\ X \cong \mathsf{B}\ (X, \mathsf{T}\ X)$ cuja base é o bifunctor B, bem como a sua propriedade universal, são representados a seguir:

$$\begin{tikzcd} \mathsf{T}\ X \arrow[d, "(\!| g |\!)"'] & \mathsf{B}\ (X, \mathsf{T}\ X) \arrow[l, "\mathsf{in}"'] \arrow[d, "\mathsf{B}\ (id, (\!| g |\!)) = \mathsf{F}\ (\!| g |\!)"] \\ B & \mathsf{B}\ (X, B) \arrow[l, "g"] \end{tikzcd} \qquad k = (\!| g |\!) \quad \equiv \quad k \cdot \mathsf{in} = g \cdot \underbrace{\mathsf{B}\ (id, k)}_{\mathsf{F}\ k}$$

(Repare-se que se tem sempre $\mathsf{F}\ k = \mathsf{B}\ (id, k)$.) Partindo da definição *genérica* de map associado ao tipo T, $\mathsf{T}\ f = (\!| \mathsf{in} \cdot \mathsf{B}\ (f, id) |\!)$ dada no formulário, mostre que o map das sequências finitas (vulg. listas) é a função

$$\begin{aligned} & f^*\ [\,] = [\,] \\ & f^*\ (h : t) = f\ h : f^*\ t \end{aligned}$$

Resolução: Listas:

$T\,X = X^*;\ f^* = map\ f;\ B(X, Y) = 1 + X \times Y,\ in = [nil, cons],\ nil\ x = [\,],\ cons(h, t) = h{:}\,t$

$$map\ f = T\,f$$

$\equiv \{\ formulário{:}\ (50)\ \}$

$$map\ f = (\!|\ in \cdot B(f, id)\ |\!)$$

$\equiv \{\ universal\ cata\ \}$

$$map\ f \cdot in = in \cdot B(f, id) \cdot F\,(map\ f)$$

$\equiv \{\ def\ de\ B\ e\ de\ F\ \}$

$$map\ f \cdot in = in \cdot (id + f \times id) \cdot B(id + id \times map\ f)$$

$\equiv \{\ def\ in;\ fusão + (à\ esquerda);\ absorção + (direita)\ \}$

$$[map\ f \cdot nil,\ map\ f \cdot cons] = [nil, cons \cdot (f \times id)] \cdot (id + id \times map\ f)$$

$\equiv \{\ absorção + (direita)\ \}$

$$[map\ f \cdot nil,\ map\ f \cdot cons] = [nil, cons \cdot (f \times id) \cdot (id \times map\ f)\,]$$

$\equiv \{\ Eq -+\ \}$

$$\begin{aligned} & map\ f \cdot nil = nil \\ & map\ f \cdot cons = cons \cdot (f \times id) \cdot (id \times map\ f) \end{aligned}$$

$\equiv \{\ Functor \times\ \}$

$$map\ f\ \cdot nil = nil$$
$$map\ f\ \cdot cons = cons \cdot (f \times map\ f)$$

$\equiv \{\ (71, 72)\ \}$

$$map\ f\ (nil\ x) = nil\ x$$
$$map\ f\ (cons(a, b)) = cons\ ((f \times map\ f)\ (a, b))$$

$\equiv \{\ 77\ \}$

$$map\ f\ (nil\ x) = nil\ x$$
$$map\ f\ (cons(a, b)) = cons\ (f\ a,\ map\ f\ b)$$

$\equiv \{\ def\ cons\ e\ de\ nil\ \}$

$$map\ f\ [] = []$$
$$map\ f\ (a : b) = f\ a : map\ f\ b$$

F09-Q2

2. Recorra à lei da absorção-cata, entre outras, para verificar as seguintes propriedades sobre listas

$$\mathsf{length} = \mathsf{sum} \cdot (\mathsf{map}\ \underline{1}) \tag{F1}$$
$$\mathsf{length} = \mathsf{length} \cdot (\mathsf{map}\ f) \tag{F2}$$

onde length, sum e map são catamorfismos de listas que conhece.

Resolução: Listas:

- $T\ X = X^*,\ T\ f = f^* = map\ f$
- $B(X, Y) = 1 + X \times Y$
- $in = [nil, cons],\ nil\ x = [],\ cons(h, t) = h : t$

$$length = (\!|\, [zero, succ \cdot \pi2]\, |\!)$$
$$sum = (\!|\, [zero, add]\, |\!)$$

(F1)

$$length = sum \cdot (map\ \underline{1})$$

$\equiv \{$ $def\ sum\ ;\ para\ listas\ T\ f\ =\ map\ f$ $\}$

$$length\ =\ (\![zero, add]\!)\ \cdot\ T\ \underline{1}$$

$\equiv \{$ $preencher$ $\}$

$$length\ =\ (\![zero, add]\ \cdot\ B(\underline{1}, id)\!)$$

$\equiv \{$ $preencher$ $\}$

$$length\ =\ (\![zero, add]\ \cdot\ (id\ +\ \underline{1}\ \times\ id)\!)$$

$\equiv \{$ $preencher$ $\}$

$$(\![zero, succ\ \cdot\ \pi 2]\!)\ =\ (\![zero, add\ \cdot\ (\underline{1}\ \times\ id)]\!)$$

$\Leftarrow \{$ (71)$\}$

$$[zero, succ\ \cdot\ \pi 2]\ =\ [zero, add\ \cdot\ (\underline{1}\ \times\ id)]$$

$\equiv \{$ Eq+ $\}$

$$succ\ \cdot\ \pi 2\ =\ add\ \cdot\ (\underline{1}\ \times\ id)$$

$\equiv \{$ $(71, 72, 77)\ por\ introdução\ da\ variável\ (a, b)$$\}$

$$succ\ b\ =\ add\ (1, b)$$

$\equiv \{$ $def\ de\ succ,\ add\ e\ função\ constante$$\}$

$$b\ +\ 1\ =\ 1\ +\ b$$

$\equiv \{$ qualquer coisa é igual a si própria $\}$

$$b\ +\ 1\ =\ 1\ +\ b$$

(F2)

$$length\ =\ length\ \cdot\ (map\ f\ )$$

$\equiv \{$ $T\ f\ =\ map\ f\ no\ caso\ das\ listas$ $\}$

$$length\ =\ (\![zero, succ\ \cdot\ \pi 2]\!)\ \cdot\ (T\ f\ )$$

$\equiv \{$ $absorção\ cata$ $\}$

$$length = (\!|\, [zero, succ \cdot \pi 2] \cdot B(f, id) \,|\!)$$

$\equiv \{$ *B das listas* $\}$

$$length = (\!|\, [zero, succ \cdot \pi 2] \cdot (id + f \times id) \,|\!)$$

$\equiv \{$ *absorção* $+$ $\}$

$$length = (\!|\, [zero, succ \cdot \pi 2 \cdot (f \times id)] \,|\!)$$

$\equiv \{$ *natural* $\pi 2$ $\}$

$$length = (\!|\, [zero, succ \cdot \pi 2] \,|\!)$$

$\equiv \{$ *definição dada de length* $\}$

true

F09-Q5

5. Considere a função $depth = (\!|\, [\mathsf{one}, \mathsf{succ} \cdot umax] \,|\!)$ que calcula a profundidade de árvores do tipo

$$\mathsf{T}\ X = \mathsf{LTree}\ X \qquad \begin{cases} \mathsf{B}\ (X, Y) = X + Y^2 \\ \mathsf{B}\ (f, g) = f + g^2 \end{cases} \qquad \mathsf{in} = [Leaf\ , Fork]$$

Haskell: **data** LTree a = $Leaf\ a \mid Fork\ (\mathsf{LTree}\ a, \mathsf{LTree}\ a)$

onde $umax\ (a, b) = max\ a\ b$. Mostre, por absorção-cata, que a profundidade de uma árvore t não é alterada quando aplica uma função f a todas as suas folhas:

$$depth \cdot \mathsf{LTree}\ f = depth \qquad \text{(F5)}$$

Resolução: Recordar aula T de 5ª feira passada:

(a) Árvores com informação de tipo A nos nós:

$$\mathsf{T} = \mathsf{BTree}\ X \qquad \begin{cases} \mathsf{B}\,(X,Y) = 1 + X \times Y^2 \\ \mathsf{B}\,(f,g) = id + f \times g^2 \end{cases} \qquad \mathsf{in} = [\underline{Empty}\ , Node]$$

Haskell: **data** BTree a = $Empty \mid Node\ (a, (\mathsf{BTree}\ a, \mathsf{BTree}\ a))$

(b) Árvores com informação de tipo A nas folhas:

$$\mathsf{T} = \mathsf{LTree}\ X \qquad \begin{cases} \mathsf{B}\,(X,Y) = X + Y^2 \\ \mathsf{B}\,(f,g) = f + g^2 \end{cases} \qquad \mathsf{in} = [Leaf\ , Fork]$$

Haskell: **data** LTree a = $Leaf\ a \mid Fork\ (\mathsf{LTree}\ a, \mathsf{LTree}\ a)$

(c) Árvores com informação nos nós e nas folhas:

$$\mathsf{T} = \mathsf{FTree}\ Z\ X \qquad \begin{cases} \mathsf{B}\,(Z,X,Y) = Z + X \times Y^2 \\ \mathsf{B}\,(h,f,g) = h + f \times g^2 \end{cases} \qquad \mathsf{in} = [Unit\ , Comp]$$

Haskell: **data** FTree $b\ a$ = $Unit\ b \mid Comp\ (a, (\mathsf{FTree}\ b\ a, \mathsf{FTree}\ b\ a))$

(d) Árvores de expressão:

$$\mathsf{T} = Expr\ Z\ X \qquad \begin{cases} \mathsf{B}\,(Z,X,Y) = Z + X \times Y^* \\ \mathsf{B}\,(h,f,g) = h + f \times \mathsf{map}\ g \end{cases} \qquad \mathsf{in} = [Var\ , Op]$$

Haskell: **data** $Expr\ v\ o = Var\ v \mid Op\ (o, [Expr\ v\ o])$

$$depth \cdot LTree\ f = depth$$

$$\equiv \{\ def\ de\ depth;\ neste\ caso\ T = LTree\ \}$$

$$(\![one, succ \cdot umax]\!) \cdot T\ f = depth$$

$$\equiv \{\ absorção - cata\ \}$$

$$(\![[one, succ \cdot umax] \cdot B(f, id)]\!) = depth$$

$$\equiv \{\ B\ das\ LTrees\ \}$$

$$(\![[one, succ \cdot umax] \cdot (f + id)]\!) = depth$$

$$\equiv \{\ absorção +\ \}$$

$$(\![one \cdot f, succ \cdot umax]\!) = depth$$

$$\equiv \{\ one = \underline{1}\ \}$$

$$(\![\underline{1} \cdot f, succ \cdot umax]\!) = depth$$

$$\equiv \{\ (3)\ \}$$

$$(\!|\, [\, \underline{1}\, , succ\ \cdot\ umax\,]\, |\!) =\ depth$$

$\equiv \{\ definição\ dada\ de\ depth\, \}$

$$true$$

F09-Q6

6. Um *anamorfismo* é um *"catamorfismo ao contrário"*, isto é, uma função $k : A \to \mathsf{T}$ tal que

$$k = \mathsf{in} \cdot \mathsf{F}\ k \cdot g \qquad \text{(F6)}$$

escrevendo-se $k = [\!(g)\!]$. Mostre que o anamorfismo de listas

$$k = [\!((id + \langle f, id \rangle) \cdot \mathsf{out}_{\mathbb{N}_0})\!] \qquad \text{(F7)}$$

descrito pelo diagrama

$$\begin{tikzcd}
\mathbb{N}_0{}^* & & 1 + \mathbb{N}_0 \times \mathbb{N}_0{}^* \arrow[ll, "\mathsf{in}"'] \\
\mathbb{N}_0 \arrow[u, "k"] \arrow[r, "\mathsf{out}_{\mathbb{N}_0}"'] & 1 + \mathbb{N}_0 \arrow[r, "id+\langle f, id\rangle"'] & 1 + \mathbb{N}_0 \times \mathbb{N}_0 \arrow[u, "id+id\times k"']
\end{tikzcd}$$

é a função

$$\begin{aligned}
&k\ 0 = [\,] \\
&k\ (n+1) = (2\ n+1) : k\ n
\end{aligned}$$

Resolução:

$$k\ = [\![\, (id\ + \langle\, f\, , id\, \rangle)\ \cdot\ out_{\mathbb{N}}\,]\!]$$

$\equiv \{\ universal - ana;\ F\, f\ =\ id\ +\ id\ \times f\ (listas)\, \}$

$$out\ \cdot\ k\ = (id\ +\ id\ \times\ k)\ \cdot\ (id\ + \langle\, f\, , id\, \rangle)\ \cdot\ out_{\mathbb{N}}$$

$\equiv \{\ Functor\ +;\ \ Absorção - x\, \}$

$$out\ \cdot\ k\ = (id\ + \langle\, f\, , k\, \rangle)\ \cdot\ out_{\mathbb{N}}$$

$\equiv \{\ isomorfismo\ in/out\ (listas)\, \}$

$$k = [nil, cons] \cdot (id + \langle f, k \rangle) \cdot out_{\mathbb{N}}$$

$\equiv \{$ $isomorfismo\ in/out\ (naturais)$ $\}$

$$k \cdot [zero, succ] = [nil, cons] \cdot (id + \langle f, k \rangle)$$

$\equiv \{$ $Fusão + à\ esquerda;\ absorção\ à\ direita$ $\}$

$$[k \cdot zero, k \cdot succ] = [nil, cons \cdot \langle f, k \rangle]$$

$\equiv \{$ $Eq +$ $\}$

$$k \cdot zero = nil$$
$$k \cdot succ = cons \cdot \langle f, k \rangle$$

$\equiv \{$ $Igualdade\ Extensional$ $\}$

$$k\ (zero\ x) = nil\ x$$
$$k\ (succ\ x) = cons\ (\langle f, k \rangle x)$$

$\equiv \{$ $preencher\ (TPC)$; f n = 2*n+1 $\}$

$$k\ 0 = []$$
$$k\ (x + 1) = (2x + 1) : k\ x$$

F09-Q5

3. A função concat, extraída do *Prelude* do Haskell, é o catamorfismo de listas

$$\mathsf{concat} = (\![\mathsf{nil}\ , \mathsf{conc}]\!) \tag{F3}$$

onde $\mathsf{conc}\ (x, y) = x +\!\!+\ y$ e $\mathsf{nil}\ \_ = [\,]$. Apresente justificações para a prova da propriedade

$$\mathsf{length} \cdot \mathsf{concat} = \mathsf{sum} \cdot \mathsf{map\ length} \tag{F4}$$

que a seguir se apresenta, onde é de esperar que as leis de *fusão*-cata e *absorção*-cata desempenhem um papel importante:

$$
\begin{array}{ll}
& \mathsf{length} \cdot \mathsf{concat} = \mathsf{sum} \cdot \mathsf{map\ length} \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mathsf{length} \cdot (\![\mathsf{nil}\ , \mathsf{conc}]\!) = (\![\underline{0}\ , \mathsf{add}]\!) \cdot \mathsf{map\ length} \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mathsf{length} \cdot \mathsf{concat} = (\![[\underline{0}\ , \mathsf{add}] \cdot (id + \mathsf{length} \times id)]\!) \\
\Leftarrow & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mathsf{length} \cdot [\mathsf{nil}\ , \mathsf{conc}] = [\underline{0}\ , \mathsf{add} \cdot (\mathsf{length} \times id)] \cdot (id + id \times \mathsf{length}) \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \left\{ \begin{array}{l} \mathsf{length} \cdot \mathsf{nil} = \underline{0} \\ \mathsf{length} \cdot \mathsf{conc} = \mathsf{add} \cdot (\mathsf{length} \times id) \cdot (id \times \mathsf{length}) \end{array} \right. \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mathsf{length} \cdot \mathsf{conc} = \mathsf{add} \cdot (\mathsf{length} \times \mathsf{length}) \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& true \\
\Box &
\end{array}
$$

Resolução:

$\equiv \{$ (1) $preencher \}$
$\equiv \{$ (2) $preencher \}$
$\Leftarrow \{$ (3) $preencher \}$
$\equiv \{$ (4) $preencher \}$
$\equiv \{$ (5) $preencher \}$
$\Leftarrow \{$ (6) $preencher \}$

[08] Aula CP/TP5 (30-Nov)

F07-Q4

4. As seguintes funções mutuamente recursivas testam a paridade de um número natural:

$$\begin{cases} impar\ 0 = \text{FALSE} \\ impar\ (n+1) = par\ n \end{cases} \qquad \begin{cases} par\ 0 = \text{TRUE} \\ par\ (n+1) = impar\ n \end{cases}$$

Assumindo o functor $\mathsf{F}\ f = id + f$, mostre que esse par de definições é equivalente ao sistema de equações

$$\begin{cases} impar \cdot \mathsf{in} = h \cdot \mathsf{F}\ \langle impar, par \rangle \\ par \cdot \mathsf{in} = k \cdot \mathsf{F}\ \langle impar, par \rangle \end{cases}$$

para um dado h e k (deduza-os). De seguida, recorra às leis da recursividade mútua e da troca para mostrar que

$$imparpar = \langle impar, par \rangle = \mathsf{for\ swap}\ (\text{FALSE}, \text{TRUE})$$

Resolução:

Parte I: $Sabemos\ que\ in\ [zero, succ]\ e\ que\ F\ f\ =\ id\ +\ f$

$$impar\ \cdot\ in\ =\ [h1, h2]\ \cdot\ (id\ +\ \langle impar, par \rangle)$$

$\equiv \{\ in\ =\ [zero, succ]\ em\ que\ succ(n)\ =\ n\ +\ 1\ e\ zero\ x\ =\ 0\ \}$

$$impar\ \cdot\ [zero, succ]\ =\ [h1, h2]\ \cdot\ (id\ +\ \langle impar, par \rangle)$$

$\equiv \{\ fusão\ +\ à\ esquerda;\ absorção\ à\ direita\ \}$

$$[impar\ \cdot\ zero,\ impar\ \cdot\ succ]\ =\ [h1, h2\ \cdot\ \langle impar, par \rangle]$$

$\equiv \{\ Eq\ +\ \}$

$impar \cdot zero = h1$

$impar \cdot succ = h2 \cdot \langle impar, par \rangle$

$\equiv$ { (71) e (72)e (76) }

$impar\ 0 = h1\ x$

$impar\ (n + 1) = h2\ (impar\ n, par\ n)$

$\equiv$ { definir $h1\ x = FALSE\ e\ h2 = \pi 2$ }

$impar\ 0 = FALSE$
$impar\ (n + 1) = par\ n$

$h = [h1, h2] = [\underline{FALSE}, \pi 2]$

Parte II:

$par \cdot in = [k1, k2] \cdot (id + \langle impar, par \rangle)$

$\equiv$ { $definição\ de\ in$ }

$par \cdot [zero, succ] = [k1, k2] \cdot (id + \langle impar, par \rangle)$

$\equiv$ { $fusão + à\ esquerda;\ absorção\ à\ direita$ }

$[par \cdot zero, par \cdot succ] = [k1, k2 \cdot \langle impar, par \rangle]$

$\equiv$ { $Eq +$ }

$par \cdot zero = k1$
$par \cdot succ = k2 \cdot \langle impar, par \rangle$

$\equiv$ { (71) e (72)e (76 }

$par\ (zero\ x) = k1\ x$
$par\ (succ\ n) = k2\ (impar\ n, par\ n)$

$\equiv$ { }

$par\ 0 = k1\ x$
$par\ (n + 1) = k2\ (impar\ n, par\ n)$

$\equiv$ { $k1 = \underline{TRUE}\ e\ k2 = \pi 1$ }

$$par\ 0 = TRUE$$
$$par\ (n+1) = impar\ n$$

$$k = [k1, k2] = [\underline{TRUE}, \pi 1]$$

Parte III

$$\begin{cases} impar \cdot in = [\underline{FALSE}, \pi 2] \cdot F\ \langle impar, par \rangle \\ par \cdot in = [\underline{TRUE}, \pi 1] \cdot F\ \langle impar, par \rangle \end{cases}$$

$\equiv \{ recursividade\ mútua\ (52) \}$

$$< impar, par > = (\!| < [\underline{FALSE}, \pi 2], [\underline{TRUE}, \pi 1] > |\!)$$

$\equiv \{\ lei\ da\ troca\ \}$

$$< impar, par > = (\!| [< \underline{FALSE}, \underline{TRUE} >, < \pi 2, \pi 1 >] |\!)$$

$\equiv \{\ < \underline{a}\ ,\ \underline{b}\ >= \underline{(a, b)}\ ver\ ficha\ 3\ \}$

$$< impar, par > = (\!| [\underline{(FALSE, TRUE)},\ swap] |\!)$$

$\equiv \{\ for\ b\ i = (\!| [\underline{i}, b] |\!)\ \}$

$$< impar, par > = for\ swap\ (FALSE, TRUE)$$

F07-Q5

5. A seguinte função em Haskell calcula a lista dos primeiros n números naturais por ordem inversa:

$$insg\ 0 = [\,]$$
$$insg\ (n+1) = (n+1) : insg\ n$$

Mostre que $insg$ pode ser definida por recursividade mútua tal como se segue,

$$insg\ 0 = [\,]$$
$$insg\ (n+1) = (fsuc\ n) : insg\ n$$
$$fsuc\ 0 = 1$$
$$fsuc\ (n+1) = fsuc\ n + 1$$

e, usando a lei de recursividade mútua, derive:

$$insg = \pi_2 \cdot insgfor$$
$$insgfor = \mathsf{for}\ \langle (1+) \cdot \pi_1, \mathsf{cons} \rangle\ (1, [\,])$$

Resolução. *Sabemos que* $in = [zero, succ]$ *e que* $F\,f = id + f$*, onde* $zero = \underline{0}$

$$\begin{cases} insg\ 0 = [] \\ insg\ (n+1) = (n+1) : insg\ n \end{cases}$$

$\equiv \{\ definir\ fsuc\ n = n + 1\ \}$

$$\begin{cases} insg\ 0 = [] \\ insg\ (n+1) = (fsuc\ n) : insg\ n \end{cases}$$

$\equiv \{\ instanciar\ \ fsuc\ n\ (para\ n := 0\ e\ n := n + 1)\}$

$$\begin{cases} insg\ 0 = [] \\ insg\ (n+1) = (fsuc\ n) : insg\ n \end{cases}$$

$$\begin{cases} fsuc\ 0 = 1 \\ fsuc\ (n+1) = fsuc\ n + 1 \end{cases}$$

$\equiv \{\ (71,72)\ ;\ cons(a, b) = a : b\ \}$

$$\begin{cases} insg \cdot \underline{0} = nil \\ insg\ (n+1) = cons(fsuc\ n, insg\ n) \end{cases}$$

$$\begin{cases} fsuc \cdot \underline{0} = \underline{1} \\ fsuc \cdot succ = succ \cdot fsuc \end{cases}$$

$\equiv \{ \; preencher \; \}$

$$\begin{cases} insg \cdot \underline{0} = nil \\ insg \cdot succ = cons \cdot < fsuc, insg > \end{cases}$$

$$\begin{cases} fsuc \cdot \underline{0} = \underline{1} \\ fsuc \cdot succ = succ \cdot fsuc \end{cases}$$

$\equiv \{ \; preencher \; \}$

$$[insg \cdot \underline{0}, insg \cdot succ] = [nil \cdot id, cons \cdot < fsuc, insg >]$$
$$[fsuc \cdot \underline{0}, fsuc \cdot succ] = [\underline{1} \cdot id, (succ \cdot \pi1) \cdot < fsuc, insg >]$$

$\equiv \{ \; fusão + \text{ à } esquerda; \; absorção + \text{ à } direita \; \}$

$$insg \cdot [\underline{0}, succ] = [nil, cons] \cdot (id + < fsuc, insg >)$$
$$fsuc \cdot [\underline{0}, succ] = [\underline{1}, succ \cdot \pi1] \cdot (id + < fsuc, insg >)$$

$\equiv \{ \; recursividade \; mútua \; \}$

$$< fsuc, insg > = (\!| < [\underline{1}, succ \cdot \pi1], [nil, cons] > |\!)$$

$\equiv \{ \; lei \; da \; troca \; \}$

$$< fsuc, insg > = (\!| [< \underline{1}, \underline{[]} >, < succ \cdot \pi1, cons >] |\!)$$

$\equiv \{ \; < \underline{a}, \underline{b} > = \underline{(a,b)} \; ver \; ficha \; 3 \; \}$

$$< fsuc, insg > = (\!| [\underline{(1, [])}, < succ \cdot \pi1, cons >] |\!)$$

$\equiv \{ \; for \; b \; i = (\!| [\underline{i}, b] |\!) \; \}$

$< fsuc\,, insg > \;=\; for \; < succ \cdot \pi 1, cons > (1, [])$

$aux(n, l) \;=\; < succ \cdot \pi 1, cons > (n, l) \;=\; ((succ \cdot \pi 1)(n, l), cons(n, l))$

$aux(n, l) = (n + 1, n : l)$

F07-Q1

1. Considere o seguinte inventário de quatro tipos de árvores:

   (a) Árvores com informação de tipo A nos nós:

   $$\mathsf{T} = \mathsf{BTree}\ A \qquad \begin{cases} \mathsf{F}\ X = 1 + A \times X^2 \\ \mathsf{F}\ f = id + id \times f^2 \end{cases} \qquad \mathsf{in} = [\underline{Empty}\ , Node]$$

   Haskell: **data** $\mathsf{BTree}\ a = Empty \mid Node\ (a, (\mathsf{BTree}\ a, \mathsf{BTree}\ a))$

   (b) Árvores com informação de tipo A nas folhas:

   $$\mathsf{T} = \mathsf{LTree}\ A \qquad \begin{cases} \mathsf{F}\ X = A + X^2 \\ \mathsf{F}\ f = id + f^2 \end{cases} \qquad \mathsf{in} = [Leaf\ , Fork]$$

   Haskell: **data** $\mathsf{LTree}\ a = Leaf\ a \mid Fork\ (\mathsf{LTree}\ a, \mathsf{LTree}\ a)$

   (c) Árvores com informação nos nós e nas folhas:

   $$\mathsf{T} = \mathsf{FTree}\ B\ A \qquad \begin{cases} \mathsf{F}\ X = B + A \times X^2 \\ \mathsf{F}\ f = id + id \times f^2 \end{cases} \qquad \mathsf{in} = [Unit\ , Comp]$$

   Haskell: **data** $\mathsf{FTree}\ b\ a = Unit\ b \mid Comp\ (a, (\mathsf{FTree}\ b\ a, \mathsf{FTree}\ b\ a))$

   (d) Árvores de expressão:

   $$\mathsf{T} = Expr\ V\ O \qquad \begin{cases} \mathsf{F}\ X = V + O \times X^* \\ \mathsf{F}\ f = id + id \times \mathsf{map}\ f \end{cases} \qquad \mathsf{in} = [Var\ , Op]$$

   Haskell: **data** $Expr\ v\ o = Var\ v \mid Op\ (o, [Expr\ v\ o])$

   Defina o gene g para cada um dos catamorfismos seguintes desenhando, para cada caso, o diagrama correspondente:

   - $zeros = (\!| g |\!)$ — substitui todas as folhas de uma árvore de tipo (1b) por zero.
   - $conta = (\!| g |\!)$ — conta o número de nós de uma árvore de tipo (1a).
   - $\mathsf{mirror} = (\!| g |\!)$ — espelha uma árvore de tipo (1b), i.e., roda-a de 180º.
   - $converte = (\!| g |\!)$ — converte árvores de tipo (1c) em árvores de tipo (1a) eliminando os Bs que estão na primeira.
   - $\mathsf{vars} = (\!| g |\!)$ — lista as variáveis de uma árvore expressão de tipo (1d).

$$
\begin{tikzcd}[column sep=huge, row sep=large]
BTree\ A \arrow[d, "conta"'] & 1 + A \times (BTree\ A)^\wedge 2 \arrow[l, "in\ =\ {[}\underline{Empty}, Node{]}"'] \arrow[d, "id\ +\ id\ \times\ (conta)^\wedge 2"] \\
N0 & 1 + A \times (N0)^\wedge 2 \arrow[l, "g\ =\ {[}g1, g2{]}"]
\end{tikzcd}
$$

$$
\begin{cases} g1 = \underline{0} \\ g2(a, (m, n)) = 1 + m + n \end{cases}
$$

$$
\begin{tikzcd}[column sep=huge, row sep=large]
LTree\ A \arrow[d, "mirror"'] & A + (LTree\ A)^\wedge 2 \arrow[l, "in\ =\ {[}Leaf, Fork{]}"'] \arrow[d, "id\ +\ (mirror)^\wedge 2"] \\
LTree\ A & A + (LTree\ A)^\wedge 2 \arrow[l, "g\ =\ {[}g1, g2{]}"]
\end{tikzcd}
$$

$$\begin{cases} g1 = Leaf \\ g2 = Fork \cdot swap \end{cases}$$

$$g = [\,Leaf \cdot id\,,\, Fork \cdot swap\,] = in \cdot (id + swap)$$

F08-Q1

1. Considere a função

$$\begin{array}{l} \mathsf{mirror}\ (Leaf\ \ a) = Leaf\ \ a \\ \mathsf{mirror}\ (Fork\ (x, y)) = Fork\ (\mathsf{mirror}\ y, \mathsf{mirror}\ x) \end{array}$$

que "espelha" árvores binárias do tipo LTree (ver fichas anteriores). Comece por mostrar que

$$\mathsf{mirror} = (\!|\, \mathsf{in} \cdot (id + \mathsf{swap})\,|\!) \tag{F1}$$

desenhando o digrama que representa este catamorfismo.
Tal como swap, mirror é um isomorfismo de árvores pois é a sua própria inversa:

$$\mathsf{mirror} \cdot \mathsf{mirror} = id \tag{F2}$$

Complete a seguinte demonstração de (F2):

$$\begin{array}{cl} & \mathsf{mirror} \cdot \mathsf{mirror} = id \\ \equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\ & \mathsf{mirror} \cdot (\!|\, \mathsf{in} \cdot (id + \mathsf{swap})\,|\!) = (\!|\, \mathsf{in}\,|\!) \\ \Leftarrow & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\ & \mathsf{mirror} \cdot \ldots = \ldots \\ \ldots & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\ & (etc) \end{array}$$

Resolução:

$$mirror \cdot mirror = id$$

$\equiv$ { $def\ mirror\ como\ catamorfismo(F1)$; $Reflexão - cata\ (47)$ }

$$mirror \cdot (\!| in \cdot (id + swap) |\!) = (\!| in |\!)$$

$\Leftarrow$ { $Fusão - cata\ (48)$ }

$$(mirror \cdot in) \cdot (id + swap) = in \cdot (id + mirror\char`^2)$$

$\equiv$ { $mirror$ é $um\ cata$: $Cancelamento - cata\ (46)$ }

$$in \cdot (id + swap) \cdot (id + mirror\char`^2) \cdot (id + swap) = in \cdot (id + mirror\char`^2)$$

$\equiv$ { $expansão\ da\ abreviatura\ mirror\char`^2$; $propriedade\ grátis\ de\ swap$ }

$$in \cdot (id + (mirror \times mirror) \cdot swap \cdot swap) = in \cdot (id + mirror \times mirror)$$

$\equiv$ { $swap \cdot swap = id$ }

$$in \cdot (id + mirror \times mirror) = in \cdot (id + mirror \times mirror)$$

$\equiv$ { $propriedade\ reflexiva\ da\ igualdade$ }

F08-Q3

$$
\begin{array}{ll}
& \mathsf{unzip}\ xs = (\mathsf{map}\ \pi_1\ xs, \mathsf{map}\ \pi_2\ xs) \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mathsf{unzip} = \langle \mathsf{map}\ \pi_1, \mathsf{map}\ \pi_2 \rangle \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mathsf{unzip} = \langle (\!| \mathsf{in} \cdot \mathsf{B}\ (\pi_1, id) |\!), (\!| \mathsf{in} \cdot \mathsf{B}\ (\pi_2, id) |\!) \rangle \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mathsf{unzip} = (\!| \langle \mathsf{in} \cdot \mathsf{B}\ (\pi_1, id) \cdot \mathsf{F}\ \pi_1, \mathsf{in} \cdot \mathsf{B}\ (\pi_2, id) \cdot \mathsf{F}\ \pi_2 \rangle |\!) \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mathsf{unzip} = (\!| \langle \mathsf{in} \cdot \mathsf{B}\ (\pi_1, \pi_1), \mathsf{in} \cdot \mathsf{B}\ (\pi_2, \pi_2) \rangle |\!) \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \mathsf{unzip} \cdot \mathsf{in} = \langle [\mathsf{nil}, \mathsf{cons} \cdot (\pi_1 \times \pi_1)], [\mathsf{nil}, \mathsf{cons} \cdot (\pi_2 \times \pi_2)] \rangle \cdot (id + id \times \mathsf{unzip}) \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \left\{ \begin{array}{l} \mathsf{unzip} \cdot \mathsf{nil} = \langle \mathsf{nil}, \mathsf{nil} \rangle \\ \mathsf{unzip} \cdot \mathsf{cons} = (\mathsf{cons} \times \mathsf{cons}) \cdot \langle \pi_1 \times \pi_1, \pi_2 \times \pi_2 \rangle \cdot (id \times \mathsf{unzip}) \end{array} \right. \\
\equiv & \quad \{ \text{ ... alguns passos mais ... } \} \\
& \ldots. \\
\equiv & \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad \} \\
& \left\{ \begin{array}{l} \mathsf{unzip}\ [\,] = ([\,], [\,]) \\ \mathsf{unzip}\ ((a, b) : xs) = ((a : as), (b : bs))\ \mathbf{where}\ (as, bs) = \mathsf{unzip}\ xs \end{array} \right.
\end{array}
$$

Justificações:

$\equiv \{$ *preencher* $\}$
$\equiv \{$ *preencher* $\}$
$\equiv \{$ *preencher* $\}$
$\equiv \{$ *preencher* $\}$
$\equiv \{$ *preencher* $\}$
$\equiv \{$ *preencher* $\}$
$\equiv \{$ *preencher* $\}$
$\equiv \{$ *preencher* $\}$
$\equiv \{$ *preencher* $\}$

---

[07] Aula CP/TP2 (23-Nov)

F06-Q2

2. Sabendo que $\mathsf{for}\ f\ i = (\![\underline{i}\ , f]\!)$ para $\mathsf{F}\ f = id + f$ (naturais), recorra à lei de fusão-cata para demonstrar a propriedade:

$$f \cdot (\mathsf{for}\ f\ i) = \mathsf{for}\ f\ (f\ i) \qquad \text{(F1)}$$

Resolução:

$$f \cdot (for\ f\ i) = for\ f\ (f\ i)$$

$\equiv \{\ for\ b\ i = (\![\underline{i}, f]\!)\ \}$

$$f \cdot (\![\underline{i}, f]\!) = (\![\underline{f\ i}, f]\!)$$

$\Leftarrow \{\ Fusão - cata\ (48)\ para\ g = [\underline{i}, f],\ h = [\underline{f\ i}, f]\ \}$

$$f \cdot [\underline{i}, f] = [\underline{f\ i}, f] \cdot (F\ f)$$

$\equiv \{\ F\ f = id + f\ \}$

$$f \cdot [\underline{i}, f] = [\underline{f\ i}, f] \cdot (id + f)$$

$\equiv \{\ Fusão -+\ (20)\ \&\ Absorção -+\ (22)\ \}$

$$[f \cdot \underline{i}, f \cdot f] = [\underline{f\ i} \cdot id, f \cdot f]$$

$\equiv \{\ Eq -+\ (27)\ \}$

$$\begin{cases} f \cdot \underline{i} = \underline{f\ i} \cdot id \\ f \cdot f = f \cdot f \end{cases}$$

$\equiv \{\ Prop.\ refl.\ da\ igualdade;\ Natural - id\ (1)\ \&\ Fusão - const\ (4)\ \}$

$$\underline{f\ i} = \underline{f\ i}$$

$\equiv \{\ Prop.\ refl.\ da\ igualdade\ \}$

$true$

F06-Q3

3. Mostre que as funções

$$f = \text{for } id \; i$$
$$g = \text{for } \underline{i} \; i$$

são a mesma função. (Qual?)

Resolução: Vamos mostrar que ambas as funções mais não são que $\underline{i}$

$$\begin{cases} for \; id \; i = \underline{i} \\ for \; \underline{i} \; i = \underline{i} \end{cases}$$

$\equiv \{ \; for \; b \; i = (\!| [\underline{i}, f] |\!) \; duas \; vezes \}$

$$\begin{cases} (\!| [\underline{i}, id] |\!) = \underline{i} \\ (\!| [\underline{i}, \underline{i}] |\!) = \underline{i} \end{cases}$$

$\Leftrightarrow \{ \; Universal - cata \; (45) \;\; duas \; vezes \}$

$$\begin{cases} \underline{i} \cdot in = [\underline{i}, id] \cdot F \; \underline{i} \\ \underline{i} \cdot in = [\underline{i}, \underline{i}] \cdot F \; \underline{i} \end{cases}$$

$\equiv \{ \; Fusão - const \; (4) \; duas \; vezes \, ; \; F \, f = id + f \; \}$

$$\begin{cases} \underline{i} = [\underline{i}, id] \cdot (id + \underline{i)} \\ \underline{i} = [\underline{i}, \underline{i}] \cdot (id + \underline{i)} \end{cases}$$

$\equiv \{ \; Absorção -+ \; (22) \; duas \; vezes; \; Natural - id \; (1) \; três \; vezes; \; Fusão - const \; (4) \}$

$$\begin{cases} \underline{i} = [\underline{i}, \underline{i}] \\ \underline{i} = [\underline{i}, \underline{i}] \end{cases}$$

$\equiv \{ \; a \wedge a = a \}$

$$\underline{i} = [\underline{i}, \underline{i}]$$

$\equiv \{ \ Universal - + \ (17) \ \}$

$$\underline{i} \cdot i1 = \underline{i}$$
$$\underline{i} \cdot i2 = \underline{i}$$

$\equiv \{ \ Fusão - const \ (4); \ Prop. \ refl. \ da \ igualdade \ \}$

$true$

$$\underline{i} = [\underline{i} \cdot id, \underline{i} \cdot id] = \underline{i} \cdot [id, id] = \underline{i}$$

F06-Q4

4. A função seguinte, em Haskell

$$sumprod \ a \ [\,] = 0$$
$$sumprod \ a \ (h : t) = a * h + sumprod \ a \ t$$

é o catamorfismo de listas

$$sumprod \ a = (\![\ [\mathsf{zero}, \mathsf{add} \cdot ((a*) \times id)] \ ]\!) \qquad \text{(F2)}$$

onde $\mathsf{zero} = \underline{0}$ e $\mathsf{add}\ (x, y) = x + y$. Mostre, como exemplo de aplicação da propriedade de **fusão-cata** para listas, que

$$sumprod \ a = (a*) \cdot \mathsf{sum} \qquad \text{(F3)}$$

onde $\mathsf{sum} = (\![\ [\mathsf{zero}, \mathsf{add}] \ ]\!)$. **NB:** não ignore propriedades elementares da aritmética que lhe possam ser úteis.

Resolução:

$$sumprod \ a = (a \ *) \cdot sum$$

$\equiv \{ \ sumprod \ a = (\![\ [zero, \ add \cdot ((a \ *) \times id)] \ ]\!) ; \ sum = (\![\ [zero, \ add \ ] \ ]\!) \ \}$

$$(\![\ [zero, \ add \cdot ((a \ *) \times id) \ ] \ ]\!) = (a \ *) \cdot (\![\ [zero, \ add \ ] \ ]\!)$$

$\Leftarrow \{ \ lei \ de \ fusão - cata \ \}$

$$(a \ *) \cdot [zero, \ add \ ] = [zero, \ add \cdot ((a \ *) \times id) \ ] \cdot F \ (a \ *)$$

$\equiv \{ \ F \ f \ para \ listas \ é \ \ id + id \times f \ \}$

$$(a\ *) \cdot [zero,\ add\,] = [zero,\ add \cdot ((a\ *) \times id)\,] \cdot (id + id \times (a\ *))$$

$\equiv$ { $Fusão\ -+$ (20) à $esquerda$ e $Absorção\ -+$ (22) à $direita;\ natural - id$ }

$$[(a\ *) \cdot zero, (a\ *) \cdot add\,] = [zero,\ add \cdot ((a\ *) \times id) \cdot (id \times (a\ *))\,]$$

$\equiv$ { $Eq\ -+$ (27) }

$$\begin{aligned} (a\ *) \cdot zero &= zero \\ (a\ *) \cdot add &= add \cdot ((a\ *) \times id) \cdot (id \times (a\ *)) \end{aligned}$$

$\equiv$ { $Functor\ -\times$ (14) ; natural id}

$$\begin{aligned} (a\ *) \cdot zero &= zero \\ (a\ *) \cdot add &= add \cdot ((a\ *) \times (a\ *)) \end{aligned}$$

$\equiv$ { $(71, 72) \times 2$; $zero\ x = 0$; $add(x, y) = x + y$ }

$$\begin{aligned} a * 0 &= 0 \\ (a\ *)(x + y) &= add(((a\ *) \times (a\ *))(x, y)) \end{aligned}$$

$\equiv$ { 77:;$add(x, y) = x + y$ }

$$\begin{aligned} a * 0 &= 0 \\ a * (x + y) &= a * x + a * y \end{aligned}$$

$\equiv$ { 0 é $absorvente\ da\ multiplicação;\ propriedade\ distributiva$ }

True

F07-Q5

5. A seguinte função em Haskell calcula a lista dos primeiros n números naturais por ordem inversa:

$$insg\ 0 = [\,]$$
$$insg\ (n+1) = (n+1) : insg\ n$$

Mostre que $insg$ pode ser definida por recursividade mútua tal como se segue,

$$insg\ 0 = [\,]$$
$$insg\ (n+1) = (fsuc\ n) : insg\ n$$
$$fsuc\ 0 = 1$$
$$fsuc\ (n+1) = fsuc\ n + 1$$

e, usando a lei de recursividade mútua, derive:

$$insg = \pi_2 \cdot insgfor$$
$$insgfor = \mathsf{for}\ \langle (1+) \cdot \pi_1, \mathsf{cons} \rangle\ (1, [\,])$$

Resolução.

$$\begin{cases} insg\ 0 = [] \\ insg\ (n+1) = (n+1) : insg\ n \end{cases}$$

$\equiv \{ \ definir\ fsuc\ n = n + 1 \ \}$

$$\begin{cases} insg\ 0 = [] \\ insg\ (n+1) = (fsuc\ n) : insg\ n \end{cases}$$

$\equiv \{ \ instanciar\ \ fsuc\ n \ \}$

$$\begin{cases} insg\ 0 = [] \\ insg\ (n+1) = (fsuc\ n) : insg\ n \end{cases}$$

$$\begin{cases} fsuc\ 0 = 1 \\ fsuc\ (n+1) = fsuc\ n + 1 \end{cases}$$

$\equiv \{$
$remoção\ de\ variáveis\ (71,\ 72,\ 77);\ definição\ das\ funções\ constantes\ nil,\ zero,\ one \ \}$

$$\begin{cases} insg \cdot zero = nil \\ insg \cdot succ = cons \cdot < fsuc , insg > \end{cases}$$

$$\begin{cases} fsuc \cdot zero = one \\ fsuc \cdot succ = succ \cdot fsuc \end{cases}$$

$\equiv$ {
$Eq +$ "$ao\ contrário$"; $natural - id$; $introdução\ do\ split\ por\ fsuc = \pi1 \cdot < fsuc, insg >$}

$$\begin{cases} [insg \cdot zero, insg \cdot succ] = [nil \cdot id , cons \cdot < fsuc , insg >] \\ [fsuc \cdot zero, fsuc \cdot succ] = [one \cdot id , succ \cdot \pi1 \cdot < fsuc, insg >] \end{cases}$$

$\equiv$ { $fusão\ -+\ (lado\ esquerdo);\ absorção\ +\ (lado\ direito)$}

$$\begin{cases} insg \cdot [zero, succ] = [nil, cons] \cdot (id + < fsuc , insg >) \\ fsuc \cdot [zero, succ] = [one , succ \cdot \pi1] \cdot (id + < fsuc, ings >) \end{cases}$$

$\equiv$ { $cancelamento\ \times$}

$$\begin{cases} insg \cdot [zero, succ] = [nil, cons] \cdot (id + < fsuc , insg >) \\ fsuc \cdot [zero, succ] = [one , succ \cdot \pi1] \cdot (id + < fsuc, insg >) \end{cases}$$

$\equiv$ { $recursividade\ mútua$ }

$$< insg, fsuc > = (\!| < [nil, cons] , [one , succ \cdot \pi1] > |\!)$$

$\equiv$ { $Lei\ da\ Troca$; $funções\ constantes$}

$$< insg, fsuc > = (\!|[< \underline{[]}, \underline{1} >, < cons, succ \cdot \pi1 >]|\!)$$

$\equiv$ { split de funções constantes }

$$< insg, fsuc > = (\!|[\underline{([], 1)} , < cons, succ \cdot \pi1 >]|\!)$$

$\equiv$ { $for\ b\ i = (\!|[\underline{i} , b]|\!)$ }

$$< fsuc, insg > \;=\; for\ < succ \cdot \pi 1, cons > (1, [])$$

---

[06] Aula CP/TP5 (16-Nov)

F05-Q5

5. Para o caso de um *isomorfismo* α, têm-se as equivalências:

$$\alpha \cdot g = h \;\equiv\; g = \alpha^{\circ} \cdot h \tag{F3}$$

$$g \cdot \alpha = h \;\equiv\; g = h \cdot \alpha^{\circ} \tag{F4}$$

Recorra a essas propriedades para mostrar que a igualdade

$$h \cdot \mathsf{distr} \cdot (g \times (id + \alpha)) = k$$

é equivalente à igualdade

$$h \cdot (g \times id + g \times \alpha) = k \cdot \mathsf{undistr}$$

(**Sugestão:** não ignore a propriedade natural (i.e. *grátis*) do isomorfismo distr.)

Resolução: Seguindo a sugestão, vamos derivar a propriedade grátis de $distr$

$$\begin{tikzcd}
1 \arrow[d, "6"'] & 2 \arrow[l, "0"'] \arrow[d, "7"] \\
3 & 4 \arrow[l, "5"]
\end{tikzcd}$$

$$(g \times h \;+\; g \times f)\,.\,distr \;=\; distr \cdot (g \times (h + f))$$

De seguida vamos usá-la no cálculo pedido:

$$h \cdot distr \cdot (g \times (id + \alpha)) = k$$

$$\equiv \{ \ propriedade\ grátis\ de\ distr,\ para\ h := id,\ f := \alpha \ \}$$

$$h \cdot (g \times id + g \times \alpha) \cdot distr = k$$

$$\equiv \{ \ (F4)\ para\ \alpha := distr\ e\ \alpha^{\text{º}} = undistr \ \}$$

$$h \cdot (g \times id + g \times \alpha) = k \cdot undistr$$

F05-Q7

7. Seja dada a seguinte codificação em Haskell do combinador

$$\begin{cases} \mathsf{for}\ b\ i\ 0 = i \\ \mathsf{for}\ b\ i\ (n+1) = b\ (\mathsf{for}\ b\ i\ n) \end{cases} \tag{F5}$$

que implementa a noção de ciclo-for, onde b é o corpo ("body") do ciclo e i é a sua inicialização.

Repetindo o processo que foi feito na aula teórica para a função $(a\times)$, calcule a partir de (F5) a função g que encaixa no diagrama seguinte,

$$\begin{tikzcd}
\mathbb{N}_0 \arrow[rr, bend left, "\mathsf{out}"] \arrow[d, "\mathsf{for}\ b\ i"'] \arrow[rr, phantom, "\cong"] & & 1 + \mathbb{N}_0 \arrow[ll, bend left, "\mathsf{in}"] \arrow[d, "id + (\mathsf{for}\ b\ i)"] \\
B & & 1 + B \arrow[ll, bend left, "g"]
\end{tikzcd}
\qquad \text{onde} \quad \begin{cases} \mathsf{in} = [\mathsf{zero}\ , \mathsf{succ}] \\ \mathsf{zero}\ _ = 0 \\ \mathsf{succ}\ n = n + 1 \end{cases}$$

Resolução: resolver $for\ b\ i \cdot in = g \cdot (id + for\ b\ i)$ em ordem a g por forma a obter a definição dada:

$$for\ b\ i \cdot in = g \cdot (id + for\ b\ i)$$

$$\equiv \{ \ in = [zero,\ succ] \ \}$$

$$for\ b\ i \cdot [zero,\ succ] = g \cdot (id + for\ b\ i)$$

$\equiv \{\ fusão\ -+ \}$

$$[for\ b\ i \cdot zero,\ for\ b\ i \cdot succ] = g \cdot (id + for\ b\ i)$$

$\equiv \{\ mudança\ de\ variável\ -\ g := [g1, g2] \}$

$$[for\ b\ i \cdot zero,\ for\ b\ i \cdot succ] = [g1, g2] \cdot (id + for\ b\ i)$$

$\equiv \{\ absorção\ -+\ no\ lado\ direito$; natural-id }

$$[for\ b\ i \cdot zero,\ for\ b\ i \cdot succ] = [g1, g2 \cdot for\ b\ i]$$

$\equiv \{ Eq\ -+\ (27) \}$

$$\begin{cases} for\ b\ i \cdot zero = g1 \\ for\ b\ i \cdot succ = g2 \cdot for\ b\ i \end{cases}$$

$\equiv \{ (71);\ (72) \}$

$$\begin{cases} for\ b\ i\ (zero\ x) = g1\ x \\ for\ b\ i\ (succ\ n) = g2(for\ b\ i\ n) \end{cases}$$

$\equiv \{ definições\ de\ zero\ e\ succ \}$

$$\begin{cases} for\ b\ i\ 0 = g1\ x \\ for\ b\ i\ (n + 1) = g2(for\ b\ i\ n) \end{cases}$$

Substituindo

$$g1\ x = i \wedge g2 = b$$

acima obtemos a definição dada:

$$\begin{cases} for\ b\ i\ 0 = i \\ for\ b\ i\ (n + 1) = b\ (for\ b\ i\ n) \end{cases}$$

Logo: $for\ b\ i\ .\ in = [\underline{i}, b] \cdot (id + for\ b\ i)$

F04-Q1

1. Os diagramas seguintes representam as **propriedades universais** que definem o combinador **catamorfismo** para dois tipos de dados — números naturais $\mathbb{N}_0$ à esquerda e listas finitas A^* à direita:

$$\begin{tikzcd} \mathbb{N}_0 \arrow[d, "(\!|\, g \,|\!)"'] & 1+\mathbb{N}_0 \arrow[l, "\mathsf{in}"'] \arrow[d, "id+(\!|\, g \,|\!)"] \\ B & 1+B \arrow[l, "g"] \end{tikzcd} \qquad \begin{tikzcd} A^* \arrow[d, "(\!|\, g \,|\!)"'] & 1+A\times A^* \arrow[l, "\mathsf{in}"'] \arrow[d, "id+id\times(\!|\, g \,|\!)"] \\ B & 1+A\times B \arrow[l, "g"] \end{tikzcd}$$

$$\begin{cases} \mathsf{in} = [\mathsf{zero}\,,\mathsf{succ}] \\ \mathsf{zero}\ \_ = 0 \\ \mathsf{succ}\ n = n+1 \end{cases} \qquad \begin{cases} \mathsf{in} = [\mathsf{nil}\,,\mathsf{cons}] \\ \mathsf{nil}\ \_ = [\,] \\ \mathsf{cons}\ (a,x) = a : x \end{cases}$$

$$k = (\!|\, g \,|\!) \;\equiv\; k\cdot\mathsf{in} = g\cdot(id+k) \qquad k = (\!|\, g \,|\!) \;\equiv\; k\cdot\mathsf{in} = g\cdot(id+id\times k)$$

$$\mathsf{for}\ b\ i = (\!|\, [\underline{i}\,, b] \,|\!) \qquad \mathsf{foldr}\ f\ u = (\!|\, [\underline{u}\,, \widehat{f}] \,|\!)$$

onde $\widehat{f}$ abrevia $\mathsf{uncurry} f$.

(a) Tendo em conta o diagrama da esquerda, codifique, em Haskell

$$(\!|\, g \,|\!) = g\cdot(id+(\!|\, g \,|\!))\cdot\mathsf{out}$$

e

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

em que out foi calculada numa ficha anterior. De seguida, codifique

$$f = \pi_2\cdot aux\ \textbf{where}\ aux = \mathsf{for}\ \langle\mathsf{succ}\cdot\pi_1, \mathsf{mul}\rangle\ (1,1)$$

e inspeccione o comportamento de f. Que função é essa?

(b) Identifique como catamorfismos de listas as funções seguintes, indicando o gene g para cada caso:[1]

i. k é a função que multiplica todos os elementos de uma lista

ii. $k = reverse$

iii. $k = \mathsf{concat}$

iv. k é a função $\mathsf{map}\ f$, para um dado $f : A \to B$

v. k é a função que calcula o máximo de uma lista de números naturais $(\mathbb{N}_0{}^*)$.

vi. $k = \mathsf{filter}\ p$ onde

$$
\begin{array}{l}
\mathsf{filter}\ p\ [\,] = [\,] \\
\mathsf{filter}\ p\ (h : t) = x \mathbin{+\!\!+} \mathsf{filter}\ p\ t \\
\quad \textbf{where}\ x = \textbf{if}\ (p\ h)\ \textbf{then}\ [h]\ \textbf{else}\ [\,]
\end{array}
$$

(b) i

$$
\begin{tikzcd}
A^* \arrow[d, "multip"'] & & 1 + A \times A^* \arrow[ll, "in\ =\ [nil, cons]"'] \arrow[d, "id\ +\ id\ \times\ multip"] \\
A & & 1 + A \times A \arrow[ll, "g\ =\ [g1, g2]"]
\end{tikzcd}
$$

$$
\begin{cases}
g1 = \underline{1} \\
g2(h, n) = h \times n
\end{cases}
$$

(b) ii

$$
\begin{tikzcd}
A^* \arrow[d, "reverse"'] & 1 + A \times A^* \arrow[l, "in\ =\ [nil,\, cons]"'] \arrow[d, "id\ +\ id\ \times reverse"] \\
A^* & 1 + A \times A^* \arrow[l, "g\ =\ [g1,\, g2]"]
\end{tikzcd}
$$

$$
\begin{cases} g1\,x = [] \\ g2(h, x) = x \ {+}{+}\ [h] \end{cases}
$$

(b) iii

$$
\begin{tikzcd}
(B^*)^* \arrow[d, "concat"'] & 1 + B^* \times (B^*)^* \arrow[l, "in\ =\ [nil,\, cons]"'] \arrow[d, "id\ +\ id\ \times concat"] \\
B^* & 1 + B^* \times B^* \arrow[l, "g\ =\ [g1,\, g2]"]
\end{tikzcd}
$$

$$
\begin{cases} g1\,x = [] \\ g2(x, y) = x \ {+}{+}\ y \end{cases}
$$

(b) iv $Assumindo\ f\colon A \rightarrow B$

$$
\begin{tikzcd}[column sep=huge, row sep=huge]
A^* \arrow[d, "map\ f"'] & 1 + A \times A^* \arrow[l, "in\ =\ [nil, cons]"'] \arrow[d, "id\ +\ id\ \times\ map\ f"] \\
B^* & 1 + A \times B^* \arrow[l, "g\ f\ =\ [g1,\ g2\ f]"]
\end{tikzcd}
$$

$$
\begin{cases} g1\ x = [] \\ g2\ \ f\ (h{:}\ x) = (f\ h : x) \end{cases}
$$

(b) v

$$
\begin{tikzcd}[column sep=huge, row sep=huge]
N_0^* \arrow[d, "maximum"'] & 1 + N0 \times N0^* \arrow[l, "in\ =\ [nil, cons]"'] \arrow[d, "id\ +\ id\ \times\ maximum"] \\
N_0 & 1 + N0 \times N0 \arrow[l, "g\ =\ [g1,\ g2]"]
\end{tikzcd}
$$

$$
\begin{cases} g1\ x = 0 \\ g2(h, m) = if\ h > m\ then\ h\ else\ m \end{cases}
$$

(b) vi

$$
\begin{tikzcd}
A^* \arrow[d, "filter\ p"'] & & 1 + A \times A^* \arrow[ll, "in\ =\ [nil, cons]"'] \arrow[d, "id\ +\ id\ \times\ filter\ p"] \\
A^* & & 1 + A \times A^* \arrow[ll, "g\ =\ [g1, g2\ p]"]
\end{tikzcd}
$$

$$
\begin{cases} g1\ x = [] \\ g2\ p\ (h, x) = if\ p\ h\ then\ h : x\ else\ x \end{cases}
$$

F06-Q2

2. Sabendo que for $f\ i = (\![\underline{i}\ , f]\!)$ para F $f = id + f$ (naturais), recorra à lei de fusão-cata para demonstrar a propriedade:

$$f \cdot (\text{for } f\ i) = \text{for } f\ (f\ i) \tag{F1}$$

Resolução:

$$f \cdot (for\ f\ i) = for\ f\ (f\ i)$$

$$\equiv \{\ for\ b\ i = (\![\underline{i}, b]\!)\ \}$$

$$f \cdot (\![\underline{i}, f]\!) = (\![\underline{f\ i}, f]\!)$$

$$\Leftarrow \{\ Fusão - cata,\ lei\ (48);\ F\ f = id + f \}$$

$$f \cdot [\underline{i}, f] = [\underline{f\ i}, f] \cdot (id + f)$$

$$\equiv \{\ Fusão\ -+,\ lei\ (20);\ Absorção\ -+,\ lei\ (22)\ \}$$

$$[f \cdot \underline{i}, f \cdot f] = [\underline{f\, i} \cdot id, f \cdot f]$$

$\equiv \{ \ Eq - +,\ lei\ (27),\ Natural - id,\ lei\ (1)\ \}$

$$\begin{cases} f \cdot \underline{i} = \underline{f\, i} \\ f \cdot f = f \cdot f \end{cases}$$

$\equiv \{ \ Fusão - const,\ lei(4);\ Propriedade\ reflexiva\ da\ igualdade\ \}$

$$true$$

F06-Q3

3. Mostre que as funções

$$f = \mathsf{for}\ id\ i$$
$$g = \mathsf{for}\ \underline{i}\ i$$

são a mesma função. (Qual?)

Resolução:

$$f = g$$

$\equiv \{ Def - g\ do\ enunciado \}$ $\qquad g = for\ \underline{i}\ i$

$$f = for\ \underline{i}\ i$$

$\equiv \{ Def - for \}$ $\qquad for\ b\ i = \llparenthesis [\underline{i}, b] \rrparenthesis$

$$f = \llparenthesis [\underline{i}, \underline{i}] \rrparenthesis$$

$\equiv \{ lei\ (F4),\ k := f\ ;\ g := [\underline{i}, \underline{i}]\ \}$ $\qquad k = \llparenthesis g \rrparenthesis \Leftrightarrow k \cdot in = g \cdot (id + k)$

$$f \cdot in = [\underline{i}, \underline{i}] \cdot (id + f)$$

$\equiv \{ \, lei\ (22),\ Absorção\ -+;$ $\qquad [g,h] \cdot (i,j) = [g \cdot i, h \cdot j]$

$\quad lei\ (1),\ Natural\ -\ id;$ $\qquad f \cdot id = id \cdot f = f$

$\quad lei\ (3),\ Natural\ -\ const \, \}$ $\qquad$ (id+f) $\underline{k} \cdot f = \underline{k}$

$$f \cdot in = [\underline{i}, \underline{i}]$$

$\equiv \{ \, Ficha\ "3"\ -\ Fusão\ -+ \, \}$ $\qquad [\underline{k}, \underline{k}] = \underline{k}$

$$f \cdot in = \underline{i}$$

$\equiv \{ \, lei\ (33),\ Shunt\ -\ left \, \}$ $\qquad h \cdot \alpha = k \equiv h = k \cdot \alpha^{\circ}$

$$f = \underline{i} \cdot out$$

$\equiv \{ \, lei\ (3),\ Natural\ -\ const \, \}$ $\qquad \underline{k} \cdot f = \underline{k}$

$$f = \underline{i}$$

F06-Q4

4. A função seguinte, em Haskell

$$sumprod\ a\ [\,] = 0$$
$$sumprod\ a\ (h : t) = a * h + sumprod\ a\ t$$

é o catamorfismo de listas

$$sumprod\ a \quad = \quad (\![\mathsf{zero}\,, \mathsf{add} \cdot ((a*) \times id)]\!) \qquad \text{(F2)}$$

onde $\mathsf{zero} = \underline{0}$ e $\mathsf{add}\ (x, y) = x + y$. Mostre, como exemplo de aplicação da propriedade de **fusão-cata** para listas, que

$$sumprod\ a \quad = \quad (a*) \cdot \mathsf{sum} \qquad \text{(F3)}$$

onde $\mathsf{sum} = (\![\mathsf{zero}\,, \mathsf{add}]\!)$. **NB:** não ignore propriedades elementares da aritmética que lhe possam ser úteis.

Resolução:

$$(a\ *) \cdot sum = sumprod\ \mathrm{a}$$

$\equiv \{ \ definições\ de\ sum\ e\ de\ sumprod\ (F1)\ como\ catamorfismos \, \}$

$$(a\ *)\ \cdot (\!| [zero, add] |\!) = (\!| [zero,\ add \cdot ((a\ *) \times id)] |\!)$$

$$\Leftarrow \{\ Fusão - cata;\ com\ f := (a\ *),\ g := [zero, add]\ e\ h := [zero, add \cdot ((a\ *) \times id]\ \}$$

$$(a\ *)\ \cdot [zero, add] = [zero,\ add \cdot ((a\ *) \times id)]\ \cdot F\,(a\ *)$$

$$\equiv \{\ F\,f\ = id + id \times f\ \}$$

$$(a\ *)\ \cdot [zero, add] = [zero,\ add \cdot ((a\ *) \times id)]\ \cdot (id + id \times\ (a\ *))$$

Fusão-+	$f \cdot [g\,, h] = [f \cdot g\,, f \cdot h]$	(20)
Absorção-+	$[g\,, h] \cdot (i + j) = [g \cdot i\,, h \cdot j]$	(22)

$$\equiv \{\ Fusão\ -+\ no\ lado\ esquerdo\ |\ Absorção\ -+\ \ no\ lado\ direito\ |\ \text{natural-id}\ \}$$

$$[(a\ *)\ \cdot zero, (a\ *) \cdot add] = [zero,\ add \cdot ((a\ *) \times id)\ \cdot (id \times\ (a\ *))]$$

$$\equiv \{\ Eq\ -+\ \}$$

$$\begin{aligned}(a\ *) \cdot zero &= zero\\ (a\ *) \cdot add &= add \cdot ((a\ *) \times id)\ \cdot (id \times\ (a\ *))\end{aligned}$$

$$\equiv \{\ Functor\ -\times\ \ |\ Natural - id\ \}$$

$$\begin{aligned}(a\ *) \cdot zero &= zero\\ (a\ *) \cdot add &= add \cdot ((a\ *) \times (a\ *))\end{aligned}$$

$$\equiv \{\ Igualdade\ Extensional\ \}$$

$$\begin{aligned}((a\ *) \cdot zero)\,x &= zero\,x\\ ((a\ *) \cdot add)\,x &= (add \cdot ((a\ *) \times (a\ *)))\,x\end{aligned}$$

$$\equiv \{\ \ Def - comp\ \}$$

$$(a\ *)\,(zero\,x) = zero\,x$$

Def-×	$(f \times g)\,(a, b) \quad = \quad (f\,a, g\,b)$	(77)

$$(a\ *)\,(add\,x) = add\,(((a\ *) \times (a\ *))\,x)$$

$$\equiv \{\ Def\ -\times\ \}$$

$$a\ *\ \ (zero\,x) = zero\,x$$

$$a * (add\,(x, y)) = add\,(a * x, a * y)$$

$\equiv$ { $add(x, y) = x + y$; $zero\,x = 0$ }

$$a * 0 = 0$$
$$a * (x + y) = a * x + a * y$$

$\equiv$ {*Elemento absorvente da* $*$ | *Propriedade distributiva da* $*$}

$$True$$

Como o que queremos mostrar é implicado por True então é True também.

$True \Rightarrow (a\ *) \cdot sum = sumprod$

---

[05] Aula CP/TP2 (09-Nov)

F04-Q5

5. Provar a igualdade $\overline{f \cdot (g \times h)} = \overline{\mathsf{ap} \cdot (id \times h)} \cdot \overline{f} \cdot g$ usando as leis das exponenciais e dos produtos.

Resolução: a estratégia é conseguir usar a lei de fusão $\overline{g \cdot (f \times id)} = \overline{g} \cdot f$ ou a propriedade universal. Vamos usar esta última, ficando a outra estratégia para casa:

$$\overline{f \cdot (g \times h)} = \overline{ap \cdot (id \times h)} \cdot \overline{f} \cdot g$$

$\equiv$ { $universal - exp\,(35)\ para\ k := \overline{ap \cdot (id \times h)} \cdot \overline{f} \cdot g$; $f := f \cdot (g \times h)$ }

$$ap \cdot ((\overline{ap \cdot (id \times h)} \cdot \overline{f} \cdot g) \times id) = f \cdot (g \times h)$$

$\equiv$ { $fusão - exp\,(38)\ para\ g := ap \cdot (id \times h)$; $f := \overline{f}$ }

$$ap \cdot ((\overline{ap \cdot (id \times h) \cdot (\overline{f} \times id)} \cdot g) \times id) = f \cdot (g \times h)$$

$\equiv \{ functor - \times \ (14) \}$

$$ap \cdot ((\overline{ap \cdot ((id \cdot \overline{f}) \times (h \cdot id))} \cdot g) \times id) = f \cdot (g \times h)$$

$\equiv \{ natural - id\,(1) \ \times 2\ vezes \}$

$$ap \cdot ((\overline{ap \cdot ((\overline{f} \cdot id) \times (id \cdot h))} \cdot g) \times id) = f \cdot (g \times h)$$

$\equiv \{ functor - \times \ (14) \}$

$$ap \cdot ((\overline{ap \cdot (\overline{f} \times id) \cdot (id \times h)} \cdot g) \times id) = f \cdot (g \times h)$$

$\equiv \{ cancelamento - exp\,(36) \}$

$$ap \cdot ((\overline{f \cdot (id \times h)} \cdot g) \times id) = f \cdot (g \times h)$$

$\equiv \{ fusão - exp\,(38)\ para\ g := \overline{f \cdot (id \times h)}\ ;\ f := g \}$

$$ap \cdot (\overline{f \cdot (id \times h) \cdot (g \times id)} \times id) = f \cdot (g \times h)$$

$\equiv \{ cancelamento - exp\,(36)\ para\ f := f \cdot (id \times h) \cdot (g \times id) \}$

$$f \cdot (id \times h) \cdot (g \times id) = f \cdot (g \times h)$$

$\equiv \{ functor - \times \ (14)\ ;\ natural - id\,(1) \}$

$$f \cdot (g \times h) = f \cdot (g \times h)$$

$\equiv$ { propriedade reflexiva da igualdade}

$$true$$

F05-Q1

1. Recorde o diagrama ao lado que foi usado nas aulas teóricas para formular a lei de absorção da exponenciação

$$\overline{g \cdot f} = g^B \cdot \overline{f}$$

onde

$$g^B = \overline{g \cdot \mathsf{ap}} \qquad \text{(F1)}$$

$$\begin{tikzcd}
D^B & D^B \times B \arrow[r, "\mathsf{ap}"] & D \\
C^B \arrow[u, "g^B"] & C^B \times B \arrow[u, "g^B \times id"] \arrow[r, "\mathsf{ap}"] \arrow[ur, "g \cdot \mathsf{ap}"] & C \arrow[u, "g"'] \\
A \arrow[u, "\overline{f}"] & A \times B \arrow[u, "\overline{f} \times id"] \arrow[ur, "f"'] &
\end{tikzcd}$$

Apresente justificações no raciocínio abaixo que pretende mostrar que g^B (que também se pode escrever exp g) é a função $g^B\ f = g \cdot f$:

$$
\begin{array}{ll}
& g^B = \overline{g \cdot \mathsf{ap}} \\
\equiv & \qquad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ \} \\
& g \cdot \mathsf{ap} = \mathsf{ap} \cdot (g^B \times id) \\
\equiv & \qquad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ \} \\
& g\ (\mathsf{ap}\ (f, b)) = \mathsf{ap}\ (g^B\ f, b) \\
\equiv & \qquad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ \} \\
& g^B\ f\ b = g\ (f\ b) \\
\equiv & \qquad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ \} \\
& g^B\ f = g \cdot f
\end{array}
$$

Resolução:

$\equiv\ (1)\ \{ universal - exp\ (35)\ para\ k := g^B\ ;\ f := g\ \cdot ap\ \}$

$\equiv\ (2)\ \{$
$igualdade\ extensional\ (71)\ para\ x := (f, b); def - comp\ (72); def - \times\ (77); def - id\ (73)$
$\}$

$\equiv\ (3)\ \{ simetria\ da\ igualdade;\ def - ap\ (82)\ \times\ 2\ vezes\ ; \}$

$\equiv\ (4)\ \{ def - comp\ (72);\ igualdade\ extensional\ (71)\ \}$

F05-Q3

3. Deduza o tipo mais geral da função $\alpha = (id + \pi_1) \cdot i_2 \cdot \pi_2$ e infira a respectiva propriedade *grátis* (*natural*) através de um diagrama.

Inferência do tipo de α: feito no quadro

Inferência da propriedade grátis:

$$\begin{tikzcd}
D + L \arrow[d, "d + l"'] & A \times (L \times \arrow[l, "\alpha"'] \arrow[d, "a \times (l \times m)"] \\
D' + L' & A' \times (L' > \arrow[l, "\alpha"]
\end{tikzcd}$$

$$(d + l) \cdot \alpha = \alpha \cdot (a \times (l \times m))$$

F05-Q6

6. Seja dada uma função ∇ da qual só sabe duas propriedades: $\nabla \cdot i_1 = id$ e $\nabla \cdot i_2 = id$. Mostre que, necessariamente, ∇ satisfaz também a propriedade natural $f \cdot \nabla = \nabla \cdot (f + f)$.

Resolução:

$$\nabla \cdot i_1 = id \;\wedge\; \nabla \cdot i_2 = id$$

$$\equiv \{\ universal + \ \}$$

$$\nabla = [id, id]$$

Inferência do tipo de ∇:

Inferência da propriedade grátis:

$$
\begin{tikzcd}
A \arrow[d, "f"'] & A + A \arrow[l, "\nabla"'] \arrow[d, "f + f"] \\
A' & A' + A' \arrow[l, "\nabla"]
\end{tikzcd}
$$

$$\nabla \cdot f = (f + f) \cdot \nabla$$

F05-Q4

4. Identifique, apoiando a sua resolução num diagrama, qual é a definição da função polimórfica α cuja propriedade natural ("grátis") é

$$(f + h) \cdot \alpha \quad = \quad \alpha \cdot (f + g \times h)$$

Resolução:

$$
\begin{tikzcd}
F + H \arrow[d, "f + h"'] & F + G \times H \arrow[l, "\alpha"'] \arrow[d, "f + g \times h"] \\
{} & {}
\end{tikzcd}
$$

$$3 \xleftarrow{\alpha} 4$$

$$(f + h) \cdot \alpha = \alpha \cdot (f + g \times h)$$

---

[04] Aula CP/TP2 (02-Nov)

F03-Q6

6. Sabendo que as igualdades

$$p \rightarrow k \; , \; k \; = \; k \tag{F4}$$
$$(p? + p?) \cdot p? \; = \; (i_1 + i_2) \cdot p? \tag{F5}$$

se verificam, demonstre as seguintes propriedades do mesmo combinador:

$$\langle (p \rightarrow f \; , \; h), (p \rightarrow g \; , \; i) \rangle \; = \; p \rightarrow \langle f, g \rangle \; , \; \langle h, i \rangle \tag{F6}$$
$$\langle f, (p \rightarrow g \; , \; h) \rangle \; = \; p \rightarrow \langle f, g \rangle \; , \; \langle f, h \rangle \tag{F7}$$
$$p \rightarrow (p \rightarrow a \; , \; b) \; , \; (p \rightarrow c \; , \; d) \; = \; p \rightarrow a \; , \; d \tag{F8}$$

Resolução da lei (F7):

$$\langle f, (p \rightarrow g, h) \rangle$$

$$= \{ \; (F4) \; \}$$

$$\langle (p \rightarrow f, f), (p \rightarrow g, h) \rangle$$

$$= \{ \; (F6) \; \}$$

$$p \rightarrow \langle f, g \rangle, \langle f, h \rangle$$

F04-Q1

1. Recorde a função

$$\mathsf{ap} : (C^B \times B) \to C$$
$$\mathsf{ap}\ (f, x) = f\ x$$

(a) Mostre, através da adição de variáveis, que a função f definida a seguir

$$f\ k = \mathsf{ap} \cdot (k \times id)$$

é a função

$$\mathsf{uncurry} :: (a \to b \to c) \to (a, b) \to c$$
$$\mathsf{uncurry}\ f\ (a, b) = f\ a\ b$$

disponível em Haskell.

(b) Mostre que a igualdade

$$\mathsf{ap} \cdot (\mathsf{curry}\ f \times id) = f \qquad \text{(F1)}$$

corresponde à definição $\mathsf{curry}\ f\ a\ b = f\ (a, b)$ da função $\mathsf{curry} :: ((a, b) \to c) \to a \to b \to c$ também disponível em Haskell.

Resolução (a):

$$f\ k = ap \cdot (k \times id)$$

$\equiv \{\ lei\ (71),\ Igualdade\ extensional\ \}$ $\qquad f = g \Leftrightarrow \langle \forall x{::}\ f\ x = g\ x \rangle$

$$(f\ k)\ x = (ap \cdot (k \times id))\ x$$

$\equiv \{\ (72);\ mudança\ de\ variável\ x := (a, b)\ \}$

$$(f\ k)\ (a, b) = ap\ ((k \times id)\ (a, b))$$

$\equiv \{\ (77)\ ;\ (1) \}$

$$(f\ k)\ (a, b) = ap\ (k\ a,\ b)$$

$\equiv \{\ ap(f, x) = f\ x\ -\ tal\ como\ no\ enunciado\ \}$

$$(f\ k)\ (a, b) = k\ a\ b$$

$\equiv \{\ definição\ de\ uncurry\ dada\ \}$

$$(f\ k)\ (a, b) = (uncurry\ k)\ (a, b)$$

$\equiv \{$ *igualdade extensional (aplicada da direita para a esquerda)* $\}$

$$f\ k\ = jose$$

$\equiv \{$ *igualdade extensional* $\}$

$$f\ = uncurry$$

Resolução (b): *parecida com (a) - TPC*

F04-Q4

4. Apresente justificações para a demonstração da igualdade

$$\overline{f}\ a = f \cdot \langle \underline{a}, id \rangle \tag{F3}$$

que se segue:

$$\begin{array}{ll}
& \overline{f}\ a = f \cdot \langle \underline{a}, id \rangle \\
\equiv & \{\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ \} \\
& \overline{f}\ a = \mathsf{ap} \cdot (\overline{f} \times id) \cdot \langle \underline{a}, id \rangle \\
\equiv & \{\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ \} \\
& \overline{f}\ a = \mathsf{ap} \cdot \langle \underline{\overline{f}\ a}, id \rangle \\
\equiv & \{\ \mathsf{ap} \cdot \langle \underline{k}, id \rangle = k \text{ (cf. } \underline{k}\ _ = k) \ \} \\
& true \\
\Box &
\end{array}$$

Resolução:

$\equiv$ (1) $\{$ (36) $\}$

$\equiv$ (2) $\{$ (11); (4); (1) $\}$

F04-Q3

3. Considere o isomorfismo de ordem superior *flip* definido pela composição de isomorfismos seguinte:

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

- Mostre que *flip*, acima definida por $\mathit{flip}\ f = \overline{\widehat{f}\cdot\mathsf{swap}}$, é um isomorfismo por ser a sua própria inversa, isto é, por

$$\mathit{flip}\ (\mathit{flip}\ f) = f \qquad \text{(F2)}$$

 se verificar.

Resolução: vamos seguir os cálculos e justificá-los com as leis da exponenciação e outras. Como é sabido das aulas teóricas, $\overline{f}$ abrevia $curry\ f$ e $\widehat{f}$ abrevia $uncurry\ f$:

$$flip\ (flip\ f)$$

$$= \{\ definição\ de\ flip\ dada\ pelo\ diagrama\ \}$$

$$flip\ (\overline{\widehat{f}\cdot swap})$$

$$= \{\ de\ novo\ definição\ de\ flip\ \}$$

$$\overline{\widehat{\overline{\widehat{f}\cdot swap}}\cdot swap}$$

$$= \{\ uncurry\ .\ curry\ =\ id\ \}$$

$$\overline{(\widehat{f}\cdot swap)\cdot swap}$$

$$= \{\ associatividade\ da\ composição\ (2)\ \}$$

$$\overline{\widehat{f}\cdot (swap\cdot swap)}$$

$$= \{\ isomorfismo\ swap\ \}$$

$$\overline{\widehat{f}\cdot id}$$

$$= \{\ natural - id\ \}$$

$$\overline{\widehat{f}}$$

$$\equiv \{ \; isomorfismo\ curry\ .\ uncurry\ =\ id \; \}$$

$$f$$

Demonstração de $flip\ f\ x\ y\ =\ f\ y\ x$:

$$flip\ f\ =\overline{\widehat{f}\cdot swap}$$

$$\equiv \{ \; k\ =\ \overline{f} \Leftrightarrow f\ =\ ap\cdot(k \times id)\ para\ k := flip\ f\ ;\ f := \widehat{f}\cdot swap \; \}$$

$$\widehat{f}\cdot swap\ =\ ap\cdot(flip\ f \times id)$$

$$\equiv \{ \; igualdade\ extensional\ (71),\ introdução\ de\ (x,y) \; \}$$

$$(\widehat{f}\cdot swap)(x,y)\ =\ (ap\cdot(flip\ f \times id))(x,y)$$

$$\equiv \{ \; (72)\text{duas vezes} \; \}$$

$$\widehat{f}(swap(x,y))\ =\ ap\ ((flip\ f \times id)(x,y))$$

$$\equiv \{ \; definição\ de\ swap;\ (77);\ (1) \; \}$$

$$\widehat{f}(y,x)\ =\ ap\ (flip\ f\ x,y)$$

$$\equiv \{ \; definição\ de\ ap \; \}$$

$$\widehat{f}(y,x)\ =\ flip\ f\ x\ y$$

$$\equiv \{ \; (84) \; \}$$

$$flip\ f\ x\ y\ =\ f\ y\ x$$

F04-Q6

6. Considere o isomorfismo

$$
\begin{tikzcd}
A^{B+C} \arrow[rr, bend left, "\mathsf{unjoin}"] \arrow[rr, phantom, "\cong"] & & A^B \times A^C \arrow[ll, bend left, "\mathsf{join}"]
\end{tikzcd}
\tag{F3}
$$

onde

$$
\begin{array}{l}
\mathsf{join}\ (f, g) = [f\ , g] \\
\mathsf{unjoin}\ k = (k \cdot i_1, k \cdot i_2)
\end{array}
$$

Mostre que $\mathsf{join} \cdot \mathsf{unjoin} = id$ e que $\mathsf{unjoin} \cdot \mathsf{join} = id$.

Resolução (a):

$$join \cdot unjoin = id$$

$$\equiv \{ \ lei\ (71),\ Igualdade\ extensional;\ lei\ (73),\ Def - id \ \}$$

$$(join \cdot unjoin)\ k = k$$

$$\equiv \{ \ (72) \ \}$$

$$join(unjoin\ k) = k$$

$$\equiv \{ \ definição\ de\ unjoin \ \}$$

$$join\ (k \cdot i1, k \cdot i2) = k$$

$$\equiv \{ \ definição\ de\ join \ \}$$

$$[k \cdot i1, k \cdot i2] = k$$

$$\equiv \{ \ universal - + \ \}$$

$$
\begin{array}{l}
k \cdot i1 = k \cdot i1 \\
k \cdot i2 = k \cdot i2
\end{array}
$$

$$\equiv \{ \ propriedade\ reflexiva\ da\ igualdade \ \}$$

$$true$$

(b):

$unjoin \cdot join = id$

$\equiv \{ (71) \}$

$unjoin(join\,(f, g)) = (f, g)$

$\equiv \{ definição\ de\ join \}$

$unjoin\,[f, g] = (f, g)$

$\equiv \{ definição\ de\ unjoin \}$

$([f, g] \cdot i1, [f, g] \cdot i2) = (f, g)$

$\equiv \{ cancelamento\ -+ \}$

$(f, g) = (f, g)$

$\equiv \{ propriedade\ reflexiva\ da\ igualmente \}$

$true$

---

[03] Aula CP/TP2 (26-Out)

F02-Q6

6. Recorra à lei Eq-+ (entre várias outras) para mostrar que a definição que conhece da função factorial,

$$\begin{array}{l} fac\ 0 = 1 \\ fac\ (n+1) = (n+1) * fac\ n \end{array}$$

é equivalente à equação seguinte

$$fac \cdot [\underline{0}, \mathsf{succ}] = [\underline{1}, \mathsf{mul} \cdot \langle \mathsf{succ}, fac \rangle].$$

onde $\mathsf{succ}\ n = n + 1$ e $\mathsf{mul}\ (a, b) = a * b$.

Resolução: estratégia mais simples é começar (sempre) pela igualdade sem variáveis e chegar à outra com variáveis. **NB**: Vamos usar as abreviaturas para facilitar a edição: $zero = \underline{0}$, $one = \underline{1}$

$$fac \cdot [zero, succ] = [one, mul \cdot \langle succ, fac \rangle]$$

$\equiv \{ fusão\ -+ (20) \}$ $\qquad f \cdot [g, h] = [f \cdot g, f \cdot h]$

$$[fac \cdot zero,\ fac \cdot succ] = [one, mul \cdot \langle succ, fac \rangle]$$

$$[f\ , g] = [h\ , k] \Leftrightarrow \left\{ \begin{array}{l} f = h \\ g = k \end{array} \right.$$

$\equiv \{ Eq\ -+\ (27) \}$

$$\begin{cases} fac \cdot (const\ 0) = const\ 1 \\ fac \cdot succ = mul \cdot \langle succ, fac \rangle \end{cases}$$

$\equiv \{ (4) \}$

$$\begin{cases} const(fac\ 0) = const\ 1 \\ fac \cdot succ = mul \cdot \langle succ, fac \rangle \end{cases}$$

$\equiv \{ regra\ da\ igualdade\ de\ funções\ constantes;\ (71)\ na\ 2^a\ linha \}$

$$\begin{cases} fac\ 0 = 1 \\ (fac \cdot succ)\ n = (mul \cdot \langle succ, fac \rangle)\ n \end{cases}$$

$\equiv \{ (72) \}$

$$\begin{cases} - \\ fac\ (succ\ n) = mul\ (\langle succ, fac \rangle\ n) \end{cases}$$

$\equiv \{ definição\ de\ succ\ dada\ ;\ (76) \}$

$$\begin{cases} -- \\ fac\ (n + 1) = mul\ (n + 1\ ,\ fac\ n) \end{cases}$$

$\equiv \{$ *definição dada*: $mul(a, b) = a * b \}$

$$\begin{cases} fac\ 0 = 1 \\ fac\ (n + 1) = (n + 1) * fac\ n \end{cases}$$

F03-Q1

1. Considere o isomorfismo

$$\begin{tikzcd} (A+B)+C \arrow[rr, bend left, "\mathsf{coassocr}"] & \cong & A+(B+C) \arrow[ll, bend left, "\mathsf{coassocl}"] \end{tikzcd}$$

onde coassocr $= [id + i_1, i_2 \cdot i_2]$. Calcule a sua conversa resolvendo em ordem a coassocl a equação,

$$\mathsf{coassocl} \cdot \mathsf{coassocr} = id$$

isto é

$$\mathsf{coassocl} \cdot \underbrace{[id + i_1, i_2 \cdot i_2]}_{\mathsf{coassocr}} = id$$

etc. Finalmente, exprima coassocl sob a forma de um programa em Haskell *não recorra* ao combinador "either".

Resolução:

$$coassocl \cdot [id + i_1, i_2 \cdot i_2] = id$$

$\equiv \{$ *lei* (20), *Fusão* $-+ \}$

$$[coassocl \cdot (id + i_1), coassocl . i_2 \cdot i_2] = id$$

$\equiv \{$
$universal\ -\!+\ \ (17),\ para\ k := \ id;\ f := coassocl \cdot (id + i_1);\ g := \ coassocl\ .\ i_2 \cdot i_2 \}$

$$id \cdot i1 = coassocl \cdot (id + i_1)$$
$$id \cdot i2 = coassocl \cdot i_2 \cdot i_2$$

$\equiv \{ definição\ f + g\ (21) \}$

$$id \cdot i1 = coassocl \cdot [i1\ .\ id,\ i2.\ i_1]$$
$$id \cdot i2 = coassocl \cdot i_2 \cdot i_2$$

$\equiv \{ fusão\ -\!+\ \ (20)$; natural-id (1) três vezes}

$$i1 = [coassocl \cdot i1,\ coassocl\ .\ i2.\ i_1]$$
$$i2 = coassocl \cdot i_2 \cdot i_2$$

$\equiv \{$
$universal\ -\!+\ \ (17)\ na\ 1ª\ linha;\ para\ k := \ i1;\ f := \ coassocl \cdot i1;\ \ g := \ coassocl\ .\ i2.\ i_1 \}$

$$i1 \cdot i1 = coassocl \cdot i1$$
$$i1 \cdot i2 = coassocl\ .\ i2.\ i_1$$
$$i2 = coassocl \cdot i_2 \cdot i_2$$

$\equiv \{ a = b$ é $a\ mesma\ coisa\ que\ b = a \}$

$$coassocl \cdot i1 = i1 \cdot i1$$
$$(coassocl \cdot i_2) \cdot i_1 = i1 \cdot i2$$
$$(coassocl \cdot i_2) \cdot i_2 = i2$$

$\equiv \{$
$universal\ -\!+\ \ (17)\ na\ 2ª\ e\ 3ª\ linha,\ para\ k := \ coassocl.\ i2;\ f := i1 \cdot i2\ ;\ \ g := \ i_2 \}$

$$coassocl \cdot i1 = i1 \cdot i1$$
$$coassocl \cdot i_2 = [i1 \cdot i2, i2]$$

$$k = [f\ , g] \quad \Leftrightarrow \quad \begin{cases} k \cdot i_1 = f \\ k \cdot i_2 = g \end{cases} \qquad (17)$$

$\equiv \{ universal\ -\!+\ \ (17),\ para\ k := \ coassocl;\ f := i1 \cdot i1\ ;\ \ g := \ [i1 \cdot i2, i2]\ \}$

$$coassocl = [i1 \cdot i1, [i1 \cdot i2, i2]\ ]$$

F03-Q2

2. O combinador

$$\begin{aligned} &\mathsf{const} :: a \to b \to a \\ &\mathsf{const}\ a\ b = a \end{aligned}$$

está disponível em Haskell para construir funções constantes, sendo habitual designarmos $\mathsf{const}\ k$ por $\underline{k}$, qualquer que seja k. Demonstre a igualdade

$$\underline{(b,a)} \quad = \quad \langle \underline{b}, \underline{a} \rangle \tag{F1}$$

a partir da propriedade universal do produto e das propriedades das funções constantes que constam do formulário.

Resolução:

$$\underline{(b,a)} = \langle \underline{b}, \underline{a} \rangle$$

$\equiv \{\ universal\ -\times\ \ (6)\ para\ k := \underline{(b,a)};\ f := \underline{b};\ g := \ \underline{a}\ \}$

$$k = \langle f, g \rangle \quad \Leftrightarrow \quad \left\{ \begin{array}{l} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{array} \right.$$

$$\left\{ \begin{array}{l} \pi 1\ .\ \underline{(b,a)} = \underline{b} \\ \pi 2\ .\ \underline{(b,a)} = \underline{a} \end{array} \right.$$

$\equiv \{\ duas\ vezes\ a\ lei\ fusão\ -\ const\ (4)\ \}$

$$\left\{ \begin{array}{l} \underline{\pi 1 (b,a)} = \underline{b} \\ \underline{\pi 2\ (b,a)} = \underline{a} \end{array} \right.$$

$\equiv \{\ igualdade\ de\ funções\ constantes\!:\ \underline{a} = \underline{b}\ \equiv\ a\ =\ b\ \}$

$$\left\{ \begin{array}{l} \pi 1 (b,a)\ =\ b \\ \pi 2 (b,a)\ =\ a \end{array} \right.$$

$\equiv \{\ definição\ das\ projeções\ (79)\ \}$

$$\left\{ \begin{array}{l} b\ =\ b \\ a\ =\ a \end{array} \right.$$

$\equiv \{\ propriedade\ reflexiva\ da\ igualdade \}$

True

F03-Q5

5. No Cálculo de Programas, as definições condicionais do tipo $h\ x = \textbf{if}\ p\ x\ \textbf{then}\ f\ x\ \textbf{else}\ g\ x$ são escritas usando o combinador ternário $p \rightarrow f\ ,\ g$ conhecido pelo nome de *condicional de McCarthy*, cuja definição

$$p \rightarrow f\ ,\ g \quad = \quad [f\ , g] \cdot p?$$

vem no formulário. Baseando-se em leis deste combinador que constam também do formulário, demostre a chamada 2ª-lei do condicional de McCarthy:

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

Resolução:

$(p \rightarrow f, g) \cdot h$

$=$ { $lei\ (30)$: $Def\ condicional\ de\ \ Mcarthy$ } $\qquad p \rightarrow f, g = [f, g] \cdot p?$

$[f, g] \cdot p? \cdot h$

$=$ { $lei\ (29)$ }

$[f, g] \cdot (h + h) \cdot (p \cdot h)?$

$=$ { $lei\ de\ absorção\ -+\ \ (22)$ } $\qquad p? \cdot f = (f + f) \cdot (p \cdot f)?$

$[f \cdot h, g \cdot h] \cdot (p \cdot h)?$

$=$ { $preencher$ } $\qquad [g, h] \cdot (i + j) = [g \cdot i, h \cdot j]$

$(p \cdot h) \rightarrow (f \cdot h), (g \cdot h)$

F03-Q6

6. Sabendo que as igualdades

$$p \to k\ ,\ k \ =\ k \tag{F4}$$
$$(p? + p?) \cdot p? \ =\ (i_1 + i_2) \cdot p? \tag{F5}$$

se verificam, demonstre as seguintes propriedades do mesmo combinador:

$$\langle (p \to f\ ,\ h), (p \to g\ ,\ i) \rangle \ =\ p \to \langle f, g \rangle\ ,\ \langle h, i \rangle \tag{F6}$$
$$\langle f, (p \to g\ ,\ h) \rangle \ =\ p \to \langle f, g \rangle\ ,\ \langle f, h \rangle \tag{F7}$$
$$p \to (p \to a\ ,\ b)\ ,\ (p \to c\ ,\ d) \ =\ p \to a\ ,\ d \tag{F8}$$

Resolução da lei (F6):

$\langle (p \to f, h), (p \to g, i) \rangle$

$= \{\ definição\ (30)\ x\ 2\ \}$ $\qquad p \to f, g\ =\ [f, g] \cdot p?$

$\langle [f, h] \cdot p?, [g, i] \cdot p?\ \rangle$

$= \{\ fusão\ -\times\ \ (9)\ \}$ $\qquad \langle g, h \rangle \cdot f\ =\ \langle g \cdot f, h \cdot f \rangle$

$\langle [f, h], [g, i] \rangle \cdot p?$

$= \{\ lei\ da\ troca\ (28)\ \}$ $\qquad [\langle f, g \rangle, \langle h, k \rangle]\ =\ \langle [f, h], [g, k] \rangle$

$[\langle f, g \rangle, \langle h, i \rangle] \cdot p?$

$=\{\ definição\ de\ condicional\ (30)\ \}$ $\qquad p \to f, g\ =\ [f, g] \cdot p?$

$p \to \langle f, g \rangle, \langle h, i \rangle$

Resolução da lei (F7) facilitada se igualarmos $f\ =\ p \to f, f$

[03] Aula CP/TP2 (26-Out)

F02-Q4

4. Seja dada a função coswap $= [i_2 , i_1]$. Faça um diagrama que explique o tipo de coswap e mostre, usando o cálculo de programas, que coswap $\cdot$ coswap $= id$.

$coswap \cdot coswap$

$= \quad \{ Def - coswap \}$ $\qquad coswap = [i_2, i_1]$

$[i_2, i_1] \cdot [i_2, i_1]$

$= \quad \{ lei\ (20),\ Fusão\ -+ \}$ $\qquad f \cdot [g, h] = [f \cdot g, f \cdot h]$

$[[i_2, i_1] \cdot i_2, [i_2, i_1] \cdot i_1]$

$= \quad \{ lei\ (18),\ Cancelamento\ -+ \}$ $\qquad \begin{cases} [f , g] \cdot i_1 = f \\ [f , g] \cdot i_2 = g \end{cases}$

$[i_1, i_2]$

$= \quad \{ lei\ (19),\ Reflexão\ -+ \}$ $\qquad [i_1, i_2] = id_{A+B}$

id

$$
\begin{tikzcd}
B \arrow[r, "i_1"] & B + A & A \arrow[l, "i_2"'] \\
A \arrow[r, "i_1"'] \arrow[urr, "id"] & A + B \arrow[u, "{[i_2, i_1]}"] & B \arrow[l, "i_2"] \arrow[ull, "id"']
\end{tikzcd}
$$

Nota: As setas diagonais que se cruzam podem parecer confusas, mas representam bem a ideia de estar um troca presente no coswap.

F02-Q6

6. Recorra à lei Eq-+ (entre várias outras) para mostrar que a definição que conhece da função factorial,

$$fac\ 0 = 1$$
$$fac\ (n+1) = (n+1) * fac\ n$$

é equivalente à equação seguinte

$$fac \cdot [\underline{0}, \mathsf{succ}] = [\underline{1}, \mathsf{mul} \cdot \langle \mathsf{succ}, fac \rangle].$$

onde succ $n = n + 1$ e mul $(a, b) = a * b$.

Resolução: estratégia mais simples é começar (sempre) pela igualdade sem variáveis e chegar à outra com variáveis. **NB**: Vamos usar as abreviaturas para facilitar a edição: $zero = \underline{0}$, $one = \underline{1}$

$$fac \cdot [zero, succ] = [one, mul \cdot \langle succ, fac \rangle]$$

$\equiv \{ \ lei\ (20),\ Fusão\ -+ \}$

$$f \cdot [g, h] = [f \cdot g, f \cdot h]$$

$$[fac \cdot zero, fac \cdot succ] = [one, mul \cdot \langle succ, fac \rangle]$$

$\equiv \{ \ lei\ (27),\ Eq\ -+\ :$

$[f\ , g] = [h\ , k] \ \Leftrightarrow \left\{ \begin{array}{l} f = h \\ g = k \end{array} \right.$

$f := fac \cdot zero,\ g := fac \cdot succ,\ h := one,\ k := mul \cdot \langle succ\ ,\ fac \rangle \}$

$$\left\{ \begin{array}{l} fac \cdot zero = one \\ fac \cdot succ = mul \cdot \langle succ, fac \rangle \end{array} \right.$$

$\equiv \{ \ lei\ (71),\ Igualdade\ extensional\ \times\ 2 \}$

$$f = g \Leftrightarrow \langle \forall x :: f\ x = g\ x \rangle$$

$$\left\{ \begin{array}{l} (fac \cdot zero)\ x = one\ x \\ (fac \cdot succ)\ x = (mul \cdot \langle succ, fac \rangle)\ x \end{array} \right.$$

$\equiv \{ \ lei\ (72),\ Def - comp\ \times\ 3 \}$ $\qquad (f \cdot g)\ x = f\ (g\ x)$

$$\left\{ \begin{array}{l} fac\ (zero\ x) = one\ x \\ fac\ (succ\ x) = mul\ (\langle succ, fac \rangle\ x) \end{array} \right.$$

$\equiv \{ \ lei\ (76),\ Def - split \}$ $\qquad \langle f, g \rangle\ x = (f\ x, g\ x)$

$$\left\{ \begin{array}{l} fac\ (zero\ x) = one\ x \\ fac\ (succ\ x) = mul\ (succ\ x, fac\ x) \end{array} \right.$$

$\equiv \{ \; Def - one, \; Def - succ, \; Def - mul, \; Def - zero \}$

$$\begin{cases} fac\,0 = 1 \\ fac\,(x + 1) = (x + 1) * fac\,x \end{cases}$$

F03-Q1

1. Considere o isomorfismo

$$\begin{tikzcd} (A+B)+C \arrow[rr, bend left, "\mathsf{coassocr}"] & \cong & A+(B+C) \arrow[ll, bend left, "\mathsf{coassocl}"] \end{tikzcd}$$

onde $\mathsf{coassocr} = [id+i_1\,, i_2 \cdot i_2]$. Calcule a sua conversa resolvendo em ordem a coassocl a equação,

$$\mathsf{coassocl} \cdot \mathsf{coassocr} \quad = \quad id$$

isto é

$$\mathsf{coassocl} \cdot \underbrace{[id + i_1\,, i_2 \cdot i_2]}_{\mathsf{coassocr}} = id$$

etc. Finalmente, exprima coassocl sob a forma de um programa em Haskell *não recorra* ao combinador "either".

Resolução:

$$coassocl \cdot [id + i_1, i_2 \cdot i_2] = id$$

$\equiv \{ \; lei\,(20), \; Fusão \; -+ \; \}$ $\qquad f \cdot [g, h] = [f \cdot g, f \cdot h]$

$$[coassocl \cdot (id + i_1), coassocl \cdot (i_2 \cdot i_2)] = id$$

$$k = [f\,, g] \quad \Leftrightarrow \quad \begin{cases} k \cdot i_1 = f \\ k \cdot i_2 = g \end{cases}$$

$\equiv \{ \; lei\,(17), \; Universal \; -+ \; :$
$\quad k = id, \; f = coassocl \cdot (id + i_1), \; g = coassocl \cdot (i_2 \cdot i_2) \; \}$

$$\begin{cases} id \cdot i_1 = coassocl \cdot (id + i_1) \end{cases}$$

$id \cdot i_2 = coassocl \cdot (i_2 \cdot i_2)$

$\equiv \{ \; lei\,(1),\; Natural - id \;\times\; 2 \; \}$ $\qquad f \cdot id = id \cdot f = f$

$$\begin{cases} i_1 = coassocl \cdot (id + i_1) \\ i_2 = coassocl \cdot (i_2 \cdot i_2) \end{cases}$$

$\equiv \{ \; lei\,(21),\; Def \; -+ \; \}$ $\qquad f + g = [i_1 \cdot f,\, i_2 \cdot g]$

$$\begin{cases} i_1 = coassocl \cdot [i_1 \cdot id, i_2 \cdot i_1] \\ i_2 = coassocl \cdot i_2 \cdot i_2 \end{cases}$$

$\equiv \{ \; lei\,(20),\; Fusão \; -+;\; lei\,(1),\; Natural - id \; \}$

$f \cdot [f, g] = [f \cdot g, f \cdot h] \qquad f \cdot id = id \cdot f = f$

$$\begin{cases} i_1 = [coassocl \cdot i_1, coassocl \cdot (i_2 \cdot i_1)] \\ i_2 = coassocl \cdot i_2 \cdot i_2 \end{cases}$$

$\equiv \{ \; lei\,(17),\; Universal \; -+ : k = i_1, f = coassocl \cdot i_1, g = coassocl \cdot (i_2 \cdot i_1) \; \}$

$$k = [f\;, g] \quad \Leftrightarrow \quad \begin{cases} k \cdot i_1 = f \\ k \cdot i_2 = g \end{cases}$$

$$\begin{cases} i_1 \cdot i_1 = coassocl \cdot i_1 \;\wedge\; i_1 \cdot i_2 = coassocl \cdot (i_2 \cdot i_1) \\ i_2 = coassocl \cdot i_2 \cdot i_2 \end{cases}$$

Daqui podemos concluir que $coassocl = [i_1 \cdot i_1, [i_1 \cdot i_2, i_2]]$, cf:

$$\begin{aligned} &coassocl\,.\; i1 = i1 \cdot i1 \\ &coassocl \cdot i2 \cdot i1 = i1 \cdot i2 \\ &coassocl \cdot i2 \cdot i2 = i2 \end{aligned}$$

$$\begin{aligned} &coassocl\,.\; i1 = i1 \cdot i1 \\ &coassocl \cdot i2 = [i1 \cdot i2,\; i2] \end{aligned}$$

$$coassocl = [i1 \cdot i1,\; i2 + id]$$

F03-Q2

2. O combinador

$$\mathsf{const} :: a \to b \to a$$
$$\mathsf{const}\ a\ b = a$$

está disponível em Haskell para construir funções constantes, sendo habitual designarmos const k por $\underline{k}$, qualquer que seja k. Demonstre a igualdade

$$\underline{(b, a)} \quad = \quad \langle \underline{b}, \underline{a} \rangle \tag{F1}$$

a partir da propriedade universal do produto e das propriedades das funções constantes que constam do formulário.

Resolução:

$$\underline{(b, a)} = \langle \underline{b}, \underline{a} \rangle$$

$\equiv \{\ lei\ (6),\ Universal\ -\times\ :\ k = \underline{(b, a)},\ f = \underline{b},\ g = \underline{a} \}$

$$k = \langle f, g \rangle \quad \Leftrightarrow \quad \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases}$$

$$\begin{cases} \pi_1 \cdot \underline{(b, a)} = \underline{b} \\ \pi_2 \cdot \underline{(b, a)} = \underline{a} \end{cases}$$

$\equiv \{\ lei\ (4),\ Fusão - const\ \}$

$$f \cdot \underline{k} = \underline{f\ k}$$

$$\begin{cases} \underline{\pi_1\ (b, a)} = \underline{b} \\ \underline{\pi_2\ (b, a)} = \underline{a} \end{cases}$$

$\equiv \{\ lei\ (79),\ Def - proj\ \}$

$$\pi_1\ (x, y) = x \ \wedge\ \pi_2\ (x, y) = y$$

$$\begin{cases} \underline{b} = \underline{b} \\ \underline{a} = \underline{a} \end{cases}$$

$\equiv \{\ propriedade\ reflexiva\ da\ igualdade\ \}$

$$True$$

F03-Q5

5. No Cálculo de Programas, as definições condicionais do tipo $h\ x = \textbf{if}\ p\ x\ \textbf{then}\ f\ x\ \textbf{else}\ g\ x$ são escritas usando o combinador ternário $p \rightarrow f\ ,\ g$ conhecido pelo nome de *condicional de McCarthy*, cuja definição

$$p \rightarrow f\ ,\ g \quad = \quad [f\ , g] \cdot p?$$

vem no formulário. Baseando-se em leis deste combinador que constam também do formulário, demostre a chamada 2ª-lei do condicional de McCarthy:

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

Resolução:

$(p \rightarrow f, g)\ \cdot\ h$

$= \{\ lei\ (30){:}\ \ Def\ condicional\ de\ \ Mcarthy\ \}$ $\qquad p \rightarrow f, g\ =\ [f, g]\ \cdot\ p?$

$preencher$

$= \{\ preencher\ \}$ $\qquad (f\ \cdot\ g)\ \cdot\ h = f\ \cdot\ (g\ \cdot\ h)$

$preencher)$

$= \{\ preencher\ \}$ $\qquad p? \cdot f\ =\ (f\ +\ f)\ \cdot\ (p\ \cdot\ f)?$

$preencher$

$= \{\ preencher\ \}$ $\qquad (f\ \cdot\ g)\ \cdot\ h = f\ \cdot\ (g\ \cdot\ h) \quad [g, h]\ \cdot\ (i\ +\ j)\ =\ [g\ \cdot\ i, h\ \cdot\ j]$

$preencher$

$= \{\ preencher\ \}$ $\qquad p \rightarrow f, g\ =\ [f, g]\ \cdot\ p?$

$(p\ \cdot\ h) \rightarrow (f\ \cdot\ h), (g\ \cdot\ h)$

F03-Q6

6. Sabendo que as igualdades

$$p \to k\ ,\ k\ =\ k \tag{F4}$$
$$(p? + p?) \cdot p?\ =\ (i_1 + i_2) \cdot p? \tag{F5}$$

se verificam, demonstre as seguintes propriedades do mesmo combinador:

$$\langle (p \to f\ ,\ h), (p \to g\ ,\ i) \rangle\ =\ p \to \langle f, g \rangle\ ,\ \langle h, i \rangle \tag{F6}$$
$$\langle f, (p \to g\ ,\ h) \rangle\ =\ p \to \langle f, g \rangle\ ,\ \langle f, h \rangle \tag{F7}$$
$$p \to (p \to a\ ,\ b)\ ,\ (p \to c\ ,\ d)\ =\ p \to a\ ,\ d \tag{F8}$$

Resolução da lei (F6):

$\langle (p \to f, h), (p \to g, i) \rangle$

$= \{\ preencher\ \}$ $\quad p \to f, g\ =\ [f, g] \cdot p?$

$preencher$

$= \{\ preencher\ \}$ $\quad \langle g, h \rangle \cdot f\ =\ \langle g \cdot f, h \cdot f \rangle$

$preencher$

$= \{\ preencher\ \}$ $\quad [\langle f, g \rangle, \langle h, k \rangle]\ =\ \langle [f, h], [g, k] \rangle$

$preencher$

$=\{\ lpreencher\ \}$ $\quad p \to f, g\ =\ [f, g] \cdot p?$

$p \to \langle f, g \rangle, \langle h, i \rangle$

Resolução da lei (F7) facilitada se igualarmos $f\ =\ p \to f, f$

[02] Aula CP/TP2 (19-Out)

F01-Q3

3. Preencha da forma mais genérica possível os "?" do diagrama
$$\begin{tikzcd} ? & ? \arrow[l, "\langle \pi_2, \pi_1 \rangle"'] & ? \arrow[l, "\langle \pi_2, \pi_1 \rangle"'] \arrow[ll, bend left, "id"] \end{tikzcd}$$

Primeiro split

$\pi_2 : A \times B \to B$
$\pi_1 : C \times D \to C$

O split vai forçar que $A \times B$seja igual a $C \times D$, quer dizer $A = C$ e $B = D$

$\pi_2 : A \times B \to B \ e \ \pi_1 : A \times B \to A$.

Agora fazemos o split:
$< \pi_2, \pi 1 > : A \times B \to B \times A$

Segundo split

$< \pi_2, \pi 1 > : C \times D \to D \times C$

Composição: a saída o 1º split $B \times A$ tem de ser igual à entrada do 2º split. Logo

$B \times A = C \times D$ ou seja, $B = C \wedge A = D$

$< \pi_2, \pi 1 > : A \times B \to B \times A$
$< \pi_2, \pi 1 > : B \times A \to A \times B$

Em suma: $< \pi_2, \pi 1 > \cdot < \pi_2, \pi 1 > : A \times B \to A \times B$

F01-Q4

4. Considere as funções seguintes:

$$f = \langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle$$
$$g = \langle id \times \pi_1, \pi_2 \cdot \pi_2 \rangle$$

Identifique os tipos de f e g. Acompanhe a sua resolução com a construção dos respectivos diagramas.

1ª parte:

$\pi_1 : A \times B \rightarrow A$
$\pi_1 : C \times D \rightarrow C$

$para\ as\ duas\ funções\ comporem:\ A = C \times D$

$\pi_1 : (C \times D) \times B \rightarrow C \times D$
$\pi_1 : C \times D \rightarrow C$

Logo $\pi_1 \cdot \pi_1 : (C \times D) \times B \rightarrow C$

$\pi_2 : E \times F \rightarrow F$
$id : G \rightarrow G$

$\pi_2 \times id : (E \times F) \times G \rightarrow F \times G$

$\pi_2 \times id : (C \times D) \times B \rightarrow D \times B$

Finalmente

$$f = < \pi_1 \cdot \pi_1, \pi_2 \times id >: (C \times D) \times B \rightarrow C \times (D \times B)$$

2ª parte:

$\pi_2 : E \times F \rightarrow F$

$id : K \rightarrow K$

Continuar

3ª parte: split força igualdade …….

Continuar

F02-Q1

1. Recorde as propriedades universais dos combinadores $\langle f, g\rangle$ e $[f, g]$,

$$k = \langle f, g\rangle \equiv \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases}$$

$$k = [f, g] \equiv \begin{cases} k \cdot i_1 = f \\ k \cdot i_2 = g \end{cases}$$

das quais, como sabe, podem ser derivadas todas as outras que aparecem no respectivo grupo, no formulário.

   (a) Use a segunda para demonstrar a lei $[i_1, i_2] = id$ conhecida por *Reflexão*-$+$.

   (b) Use a primeira para demonstrar a lei

   $$\langle h, k\rangle \cdot f = \langle h \cdot f, k \cdot f\rangle$$

   que também consta desse formulário sob a designação *fusão*-$\times$.

Resolução (a):

$[i_1, i_2] = id$

$$k = [f, g] \Leftrightarrow \begin{cases} k \cdot i_1 = f \\ k \cdot i_2 = g \end{cases}$$

$\equiv \{ \text{lei universal } -+, \text{ para } k := id,\ f := i1,\ g := i2 \}$

$$\begin{cases} id \cdot i1 = i1 \\ id \cdot i2 = i2 \end{cases}$$

$\equiv \{ \, id \text{ é } elemento\ neutro\ da\ composição \, \}$ $\qquad f \cdot id = id \cdot f = f$

$$\begin{cases} i1 = i1 \\ i2 = i2 \end{cases}$$

$\equiv \{ \, propriedade\ reflexiva\ da\ igualdade \ \times 2 \, \}$

$$True$$

Resolução (b):

$$\langle h, k \rangle \cdot f = \langle h \cdot f, k \cdot f \rangle$$

$\equiv \{ \, universal \ -\times \, \}$

$$\begin{cases} (\pi 1 . < h, k >) \cdot f = h \cdot f \\ (\pi 2 . < h, k >) \cdot f = k \cdot f \end{cases}$$

$\equiv \{ \, cancelamento \ \times \, \}$

$$\begin{cases} h \cdot f = h \cdot f \\ k \cdot f = k \cdot f \end{cases}$$

$\equiv \{ \, propriedade\ reflexiva\ da\ igualdade \ \times 2 \, \}$

$$\begin{cases} True \\ True \end{cases}$$

$\equiv \{ \, preencher \, \}$

$$True$$

F02-Q2

2. Uma função diz-se *constante* sempre que o seu resultado é o mesmo, qualquer que seja o argumento. Por isso se designa uma tal função sublinhando o valor do seu resultado: se este for k, por exemplo, ter-se-á a função $\underline{k} : A \to K$, para k um valor de K, que satisfaz sempre a propriedade

$$\underline{k} \cdot f = \underline{k}$$

qualquer que seja k e f.[1]

Mostre que $[\underline{k}, \underline{k}] = \underline{k}$ aplicando a segunda lei universal dada acima.

Resolução:

$$[\underline{k}, \underline{k}] = \underline{k}$$

$\equiv \{ \, lei\ (17),\ Universal\ -+,\ k := \underline{k};\ f := \underline{k};\ g := \underline{k} \, \}$

$$k = [f\ , g] \quad \Leftrightarrow \quad \begin{cases} k \cdot i_1 = f \\ k \cdot i_2 = g \end{cases}$$

$$\begin{cases} \underline{k} \cdot i_1 = \underline{k} \\ \underline{k} \cdot i_2 = \underline{k} \end{cases}$$

$\equiv \{ \, lei\ (3),\ Natural\ -\ const \, \}$ $\qquad \underline{k} \cdot f = \underline{k}$

$$\begin{cases} \underline{k} = \underline{k} \\ \underline{k} = \underline{k} \end{cases}$$

$\equiv \{ \, em\ matemática,\ \forall a ::\ a\ =\ a;\ a\ igualdade\ é\ uma\ relação\ reflexiva \, \}$

$$True$$

Tipos:

$$const\ k :\ A\ \to K$$

$$const\ k :\ B \to K$$

$$const\ k :\ A\ +\ B \to K$$

[01] Aula CP/TP2 (12-Out)

F01-Q1

1. A composição de funções define-se, em Haskell, tal como na matemática:

   $$(f \cdot g)\ x = f\ (g\ x)$$

   (a) Calcule $(f \cdot g)\ x$ para os casos seguintes:

   $$\begin{cases} f\ x = 2 * x \\ g\ x = x + 1 \end{cases} \quad \begin{cases} f = \mathsf{succ} \\ g\ x = 2 * x \end{cases} \quad \begin{cases} f = \mathsf{succ} \\ g = \mathsf{length} \end{cases} \quad \begin{cases} g\ (x, y) = x + y \\ f = \mathsf{succ} \cdot (2*) \end{cases}$$

   Anime as composições funcionais acima num interpretador de Haskell.

   (b) Mostre que $(f \cdot g) \cdot h = f \cdot (g \cdot h)$, quaisquer que sejam f, g e h.

   (c) A função $id :: a \to a$ é tal que $id\ x = x$. Mostre que $f \cdot id = id \cdot f = f$ qualquer que seja f.

(a-1)

$(f \ \cdot \ g)\ x$

= { definição ao lado } $\qquad (f \ \cdot \ g)\ x \ = \ f\ (g\ x)$

$f(g\ x)$

= { g x = x +1 }

$f(x\ +\ 1)$

= { f x = 2 * x }

$2\ *\ (x\ +\ 1)$

= { preencher }

$2\,x\ +\ 2$

(a-2)

$$(f \cdot g)\,x$$

= { definição ao lado } $\quad (f \cdot g)\,x = f\,(g\,x)$

$$f(g(x))$$

= { g x = 2*x }

$$f(2 * x)$$

= { f x = succ x }

$$succ(2 * x)$$

= { succ x = x+1 }

$$2x + 1$$

(a-3)

$$(f \cdot g)\,x$$

= { definição ao lado } $\quad (f \cdot g)\,x = f\,(g\,x)$

$$f(g(x))$$

= { g x = length x }

$$f(length\,x)$$

= { f x = succ x }

$$succ(length\,x)$$

= { succ x = x+1 }

$$(length\,x) + 1$$

(a-4)

$(f \cdot g)\,(x, y)$

$=$ { definição ao lado } $\qquad (f \cdot g)\,x = f\,(g\,x)$

$f(g(x, y))$

$=$ { g (x,y) = x+y }

$f(x + y)$

$=$ { f x = succ ·(2*) }

$(succ \cdot (2\,*))(x + y)$

$=$ { definição ao lado }

$succ((2\,*)(x + y))$

$=$ { succ x = x+1 }

$(2\,*)(x + y) + 1$

$=$ { Propriedade distributiva }

$2x + 2y + 1$

(b) Mostre que $(f \cdot g) \cdot h = f \cdot (g \cdot h)$, quaisquer que sejam f, g e h.

(b)

$(f \cdot g) \cdot h = f \cdot (g \cdot h)$

$\equiv$ { igualdade extensional } $\qquad f = g \Leftrightarrow \langle \forall x :: f\,x = g\,x \rangle$

$((f \cdot g) \cdot h)\,x = (f \cdot (g \cdot h))\,x$

$\equiv \{ definição\ de\ composição\ x\ 2 \}$ $\qquad (f \cdot g)\,x = f\,(g\,x)$

$(f \cdot g)\,(h\,x) = f\,((g \cdot h)\,x)$

$\equiv \{ definição\ de\ composição\ x\ 2 \}$ $\qquad (f \cdot g)\,x = f\,(g\,x)$

$f\,(g\,(h\,x)) = f\,(g\,(h\,x))$

$\equiv$ { propriedade reflexiva da igualdade, $\forall a :: \ a = \ a$ - “uma coisa é sempre igual a si própria” }

$$True$$

(c)

$$f \ \cdot id = id \cdot f = f$$

= { Lei 71 x 3 }

$$(f \ \cdot id)\, x = (id \cdot f)\, x = f\, x$$

= { (Lei 72) }

$$f(id\, x) = id\, (f\, x) = f\, x$$

= { (Lei 73) }

$$f\, x = f\, x = f\, x$$

= {}

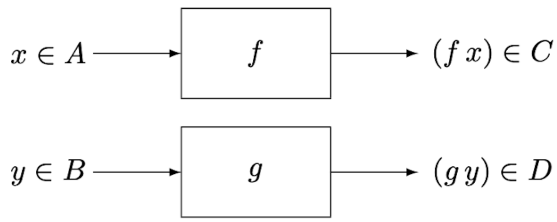
$$f = f = f$$

= { propriedade reflexiva da igualdade}

$$true$$

F01-Q2

2. O diagrama de blocos



descreve o combinador funcional *produto*

$$f \times g = \langle f \cdot \pi_1, g \cdot \pi_2 \rangle \tag{F1}$$

a) Mostre que $(f \times g)\,(x, y) = (f\,x,\ g\,y)$

$(f \times g)\,(x, y)$

$= \{\ f \times g = \langle f \cdot \pi_1, g \cdot \pi_2 \rangle$ ie. (F1) acima $\}$

$\langle f \cdot \pi_1, g \cdot \pi_2 \rangle\ (x, y)$

$= \{$ definição de split de funções $\}$

$((f \cdot \pi_1)\,(x, y)\,,\ (g \cdot \pi_2)\ (x, y))$

$= \{$ definição de composição x 2 $\}$

$(f\,(\pi_1\,(x, y))\,,\ g(\pi_2\ (x, y)))$

$= \{$ definição das projeções $\}$

$(f\,x\,,\ g\,y)$

b) Mostre que $\pi_1 \cdot (f \times g) = f \cdot \pi_1$

$$\begin{tikzcd} C & C \times D \arrow[l, "\pi_1"'] \\ A \arrow[u, "f"] & A \times B \arrow[l, "\pi_1"] \arrow[u, "f \times g"'] \end{tikzcd}$$

TPC $\pi_1 \cdot (f \times g)$

$= \{$ Def - x (10) $\}$

$\pi_1 \cdot \langle f \cdot \pi_1 , g \cdot \pi_2 \rangle$

$= \{$ Cancelamento - x (7) $\}$

$f \cdot \pi_1$

Mostre que $\pi_2 \cdot (f \times g) = g \cdot \pi_2$ (TPC)

$$\begin{tikzcd} C \times D \arrow[r, "\pi_2"] & D \\ A \times B \arrow[u, "f \times g"] \arrow[r, "\pi_2"'] & B \arrow[u, "g"'] \end{tikzcd}$$

$\pi_2 \cdot (f \times g)$

$= \{$ Def - x (10) $\}$

$\pi_2 \cdot \langle f \cdot \pi_1 , g \cdot \pi_2 \rangle$

$= \{$ Cancelamento - x (7) $\}$

$g \cdot \pi_2$

Mostre que $id \times id = id$

$$\begin{tikzcd} A & A \times B \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & B \\ A \arrow[u, "id"] & A \times B \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] \arrow[u, "id_{A \times B}"] & B \arrow[u, "id"'] \end{tikzcd}$$

$id \times id$

$= \{$ Def - x (10) $\}$

$\langle id \cdot \pi_1 , id \cdot \pi_2 \rangle$

$= \{$ Natural - id (1) $\}$

$\langle \pi_1 , \pi_2 \rangle$

$= \{$ Reflexão - x (8) $\}$

id

Mostre que $(f \times g) \cdot (h \times k) = (f \cdot h) \times (g \cdot k)$

$$(f \times g) \cdot (h \times k) = (f \cdot h) \times (g \cdot k)$$

$\equiv$ { preencher }

$$((f \times g) \cdot (h \times k))\,(x, y) = ((f \cdot h) \times (g \cdot k))(x, y)$$

$\equiv$ { esquerda: composição de funções ; direita: definição de produto }

$$(f \times g)((h \times k)\,(x, y)) = ((f \cdot h)\,x, (g \cdot k)\,y)$$

$=$ { esquerda: definição de produto; direita: composição de funções x 2 }

$$(f \times g)(h\,x,\ k\,y) = (f(h\,x), g(k\,y))$$

$=$ { esquerda: definição de produto }

$$(f(h\,x),\ g(k\,y)) = (f(h\,x), g(k\,y))$$

$=$ { propriedade reflexiva da igualdade }

$$True$$

F01-Q3

3. Preencha da forma mais genérica possível os “?” do diagrama
$$\begin{tikzcd} ? & ? \arrow[l, "\langle \pi_2, \pi_1 \rangle"'] & ? \arrow[l, "\langle \pi_2, \pi_1 \rangle"'] \arrow[ll, bend left, "id"] \end{tikzcd}$$

Primeiro split

$\pi_2 : A \times B \to B$
$\pi_1 : C \times D \to C$

Fazer split destas funções, $\langle \pi_2, \pi_1 \rangle$, origina a unificação $A \times B = C \times D$. ie $A = C \wedge B = D$

$\pi_2 : A \times B \to B$
$\pi_1 : A \times B \to A$

$\langle \pi_2, \pi_1 \rangle : A \times B \to B \times A$ **+**

Segundo split

$\langle \pi_2, \pi_1 \rangle : C \times D \to D \times C$

Logo teremos:

$\langle \pi_2, \pi_1 \rangle : A \times B \to B \times A$
$\langle \pi_2, \pi_1 \rangle : C \times D \to D \times C$

O tipo de entrada da função consumidora tem que ser igual ao tipo de saída da função produtora.

$B \times A = C \times D$ **isto é** $B = C \wedge A = D$

$\langle \pi_2, \pi_1 \rangle : A \times B \to B \times A$
$\langle \pi_2, \pi_1 \rangle : B \times A \to A \times B$

E finalmente:

$$\begin{tikzcd} A \times B & B \times A \arrow[l, "\langle \pi_2, \pi_1 \rangle"'] & A \times B \arrow[l, "\langle \pi_2, \pi_1 \rangle"'] \arrow[ll, bend left, "id"] \end{tikzcd}$$

$\langle \pi_2, \pi_1 \rangle \cdot \langle \pi_2, \pi_1 \rangle : A \times B \to A \times B$

F01-Q4

4. Considere as funções seguintes:

$$f = \langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle$$
$$g = \langle id \times \pi_1, \pi_2 \cdot \pi_2 \rangle$$

Identifique os tipos de f e g. Acompanhe a sua resolução com a construção dos respectivos diagramas.

1ª parte:

$\pi_1 : A \times B \to A$
$\pi_1 : C \times D \to C$

Unificação implicada pela composição: $C = A \times B$

$\pi_1 : A \times B \to A$
$\pi_1 : (A \times B) \times D \to A \times B$

Agora as duas funções já compõem: qual vai ser o tipo da composição?

$\pi_1 \cdot \pi_1 : (A \times B) \times D \to A$

2ª parte:

$\pi_2 : E \times F \to F$
$id : K \to K$

$\pi_2 \times id : (E \times F) \times K \to F \times K$

3ª parte: split força igualdade $(A \times B) \times D = (E \times F) \times K$ isto é $D = K$ e $A = E$ e $B = F$

$\pi_1 \cdot \pi_1 : (A \times B) \times D \to A$
$\pi_2 \times id : (A \times B) \times D \to B \times D$

$\langle \pi_1 \cdot \pi_1, \pi_2 \times id \rangle : (A \times B) \times D \to A \times (B \times D)$

F01-Q5

5. Sabe-se que uma dada função g satisfaz a propriedade:

$$(id \times \pi_2) \cdot \langle id \times \pi_2, id \times \pi_1 \rangle \cdot g = id \qquad \text{(F6)}$$

Sem calcular ou conjecturar a sua definição, determine o tipo mais geral de g completando o diagrama:

$$\begin{tikzcd}
A \times (C \times B) \arrow[d, "\langle id \times \pi_2, id \times \pi_1 \rangle"'] & \cdots \arrow[l, "g"'] \arrow[d, "id"] \\
\cdots \arrow[r, "id \times \pi_2"'] & \cdots
\end{tikzcd}$$

TPC

F01-Q6

6. Apresente definições em Haskell das seguintes funções que estudou em PF:

uncurry $:: (a \to b \to c) \to (a, b) \to c$ (que emparelha os argumentos de uma função)
curry $:: ((a, b) \to c) \to a \to b \to c$ (que faz o efeito inverso da anterior)
flip $:: (a \to b \to c) \to b \to a \to c$ (que troca a ordem dos argumentos de uma função)

```haskell
uncurry :: (a -> b -> c) -> (a, b) -> c
uncurry f (a,b) = f a b

curry :: ((a, b) -> c) -> a -> b -> c
curry g a b = g(a,b)

flip :: (a -> b -> c) -> b -> a -> c
flip f b a = f a b
```
