Cálculo de Programas
Algebra of Programming

UNIVERSIDADE DO MINHO
Lic. em Engenharia Informática (3º ano)
Lic. Ciências da Computação (2º ano)

2024/25 - Ficha *(Exercise sheet)* nr. 7

O quadro abaixo representa a **propriedade universal** que define o combinador **catamorfismo**, com duas instâncias — números naturais $\mathbb{N}_0$ e listas finitas A^*, onde $\widehat{f}$ abrevia $\mathsf{uncurry}\, f$.

The table below depicts the ***universal property*** *that defines the* ***catamorphism*** *combinator, with two instances — natural numbers $\mathbb{N}_0$ and finite lists A^*, where $\widehat{f}$ abbreviates* $\mathsf{uncurry}\, f$*:*

Catamorfismo *(Catamorphism)*:

$$\begin{tikzcd}
\mathsf{T} \arrow[rr, bend left, "\mathsf{out}"] \arrow[d, "k"'] \arrow[rr, phantom, "\cong"] & & \mathsf{F}\,\mathsf{T} \arrow[ll, bend left, "\mathsf{in}"] \arrow[d, "\mathsf{F}\,k"] \\
B & & \mathsf{F}\,B \arrow[ll, bend left, "g"]
\end{tikzcd}$$

$$k = (\!|\, g \,|\!) \;\Leftrightarrow\; k \cdot \mathsf{in} = g \cdot \mathsf{F}\; k \qquad \text{(F1)}$$

Listas *(Lists)*:

$$\left\{\begin{array}{l}
\mathsf{T} = A^* \\
\left\{\begin{array}{l} \mathsf{in} = [\mathsf{nil}\,, \mathsf{cons}] \\ \mathsf{nil}\; _ = [\,] \\ \mathsf{cons}\;(h, t) = h : t \end{array}\right. \\
\left\{\begin{array}{l} \mathsf{F}\; X = 1 + A \times X \\ \mathsf{F}\; f = id + id \times f \end{array}\right.
\end{array}\right. \qquad \mathsf{foldr}\; f\; i = (\!|\, [\underline{i}\,, \widehat{f}\,] \,|\!)$$

Números naturais *(Natural numbers)*:

$$\left\{\begin{array}{l}
\mathsf{T} = \mathbb{N}_0 \\
\left\{\begin{array}{l} \mathsf{in}_{\mathbb{N}_0} = [\underline{0}\,, \mathsf{succ}] \\ \mathsf{succ}\; x\; n = n + 1 \end{array}\right. \\
\left\{\begin{array}{l} \mathsf{F}\; X = 1 + X \\ \mathsf{F}\; f = id + f \end{array}\right.
\end{array}\right. \qquad \mathsf{for}\; b\; i = (\!|\, [\underline{i}\,, b] \,|\!)$$

1. Fazendo $\mathsf{T} = \mathbb{N}_0$, codifique — recorrendo à biblioteca Cp.hs e à definição de out feita numa ficha anterior — o combinador:

   Taking $\mathsf{T} = \mathbb{N}_0$, encode — loading the Cp.hs library and using out *defined in a previous exercise sheet, the combinator:*

$$(\!|\, g \,|\!) = g \cdot (id + (\!|\, g \,|\!)) \cdot \mathsf{out} \qquad \text{(F2)}$$

   De seguida implemente e teste a seguinte função:

   Then implement and test de following function:

$$rep\;\; a = (\!|\, [\mathsf{nil}\;, (a{:})] \,|\!) \qquad \text{(F3)}$$

O que faz ela?

What is its purpose?

---

2. Identifique como catamorfismos de listas ($k = (\!| g |\!)$) as funções seguintes, indicando o gene g para cada caso (apoie a sua resolução com diagramas):

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

   (b) $k = \mathsf{reverse}$

   (c) $k = \mathsf{concat}$

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

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

   (f) $k = \mathsf{filter}\ p$ onde:

   Identify as list catamorphisms ($k = (\!| g |\!)$) the following functions, indicating the corresponding 'gene' g for each case (support your answer with diagrams):

   (a) k is the function that multiplies all elements of a list.

   (b) $k = \mathsf{reverse}$

   (c) $k = \mathsf{concat}$

   (d) k is the function $\mathsf{map}\ f$, for a given $f : A \to B$.

   (e) k is the function that calculates the maximum of a list of natural numbers ($\mathbb{N}_0{}^*$).

   (f) $k = \mathsf{filter}\ p$ where:

$$\mathsf{filter}\ p\ [\,] = [\,]$$
$$\mathsf{filter}\ p\ (h : t) = x +\!\!+\ \mathsf{filter}\ p\ t\ \mathbf{where}\ x = \mathbf{if}\ (p\ h)\ \mathbf{then}\ [h]\ \mathbf{else}\ [\,]$$

---

3. A função seguinte, em Haskell

   The following function, in Haskell

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

é o catamorfismo de listas

is the list-catamorphism

$$sumprod\ a \quad = \quad (\!| [\mathsf{zero}\ , \mathsf{add} \cdot ((a*) \times id)] |\!) \tag{F4}$$

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

*where $\mathsf{zero} = \underline{0}$ and $\mathsf{add}\ (x, y) = x + y$. As an example of application of **cata-fusion**, prove the equality*

$$sumprod\ a \quad = \quad (a*) \cdot \mathsf{sum} \tag{F5}$$

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

*where $\mathsf{sum} = (\!| [\mathsf{zero}\ , \mathsf{add}] |\!)$. **NB:** take into account elementary arithmetic properties that may be useful.*

---

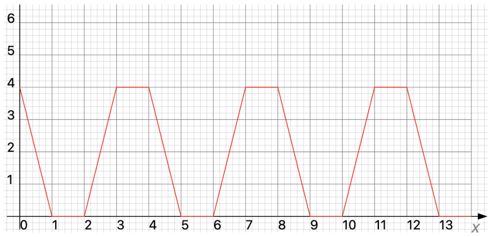
4. A função $\mathsf{foldr}\ \overline{\pi_2}\ i$ é necessariamente uma função constante. Qual? Justifique com o respectivo cálculo.

   Function $\mathsf{foldr}\ \overline{\pi_2}\ i$ is a constant function, for any i – which constant function? Write down your calculations.

---

5. A figura representa a função $\pi_1 \cdot aux$, para aux definida ao lado:

   The figure plots $\pi_1 \cdot aux$, for aux defined aside:



$$
\begin{array}{l}
aux = \mathsf{for}\ loop\ (4,-2)\ \textbf{where} \\
\quad loop\ (a,b) = (2+b, 2-a)
\end{array}
$$

Partindo da definição do combinador $\mathsf{for}\ b\ i = (\!|\,[\underline{i}\ ,b]\,|\!)$, para $\mathsf{F} = id + f$ e $\mathsf{in} = [\underline{0}\ ,\mathsf{succ}]$, resolva em ordem a f e g a equação

Starting from the definition of $\mathsf{for}\ b\ i = (\!|\,[\underline{i}\ ,b]\,|\!)$, *for* $\mathsf{F} = id + f$ *and* $\mathsf{in} = [\underline{0}\ ,\mathsf{succ}]$, *solve for f and g the equation*

$$\langle f, g\rangle = aux$$

por aplicação da lei de recursividade mútua, entregando as definições de f e g em notação *pointwise*.

by the mutual recursion law, delivering the definitions of f and g in pointwise *notation.*

---

6. Mostre que a lei da recursividade mútua generaliza a mais do que duas funções, neste caso três:

   Show that the mutual recursion law generalizes to more than two functions (three, in the following case):

$$
\left\{
\begin{array}{l}
f \cdot \mathsf{in} = h \cdot \mathsf{F}\ \langle\langle f, g\rangle, j\rangle \\
g \cdot \mathsf{in} = k \cdot \mathsf{F}\ \langle\langle f, g\rangle, j\rangle \\
j \cdot \mathsf{in} = l \cdot \mathsf{F}\ \langle\langle f, g\rangle, j\rangle
\end{array}
\right.
\quad \equiv \quad \langle\langle f, g\rangle, j\rangle = (\!|\,\langle\langle h, k\rangle, l\rangle\,|\!) \tag{F6}
$$

---

7. Considere o functor

   Consider functor

$$
\left\{
\begin{array}{l}
\mathsf{T}\ X = X \times X \\
\mathsf{T}\ f = f \times f
\end{array}
\right.
$$

e as funções

and functions

$$
\begin{array}{l}
\mu = \pi_1 \times \pi_2 \\
u = \langle id, id\rangle.
\end{array}
$$

(a) Mostre que T é de facto um functor:

(a) Show that T is indeed a functor:

$$\mathsf{T}\ id = id \tag{F7}$$

$$\mathsf{T}\ (f \cdot g) = \mathsf{T}\ f \cdot \mathsf{T} \cdot g \tag{F8}$$

(b) Demonstre a propriedade:

(b) Prove the following property:

$$\mu \cdot \mathsf{T}\ u = id = \mu \cdot u$$

---

8. **Questão prática** — Este problema não irá ser abordado em sala de aula. Os alunos devem tentar resolvê-lo em casa e, querendo, publicarem a sua solução no canal **#geral** do Slack, com vista à sua discussão com colegas.

   Open assignment *— This assignment will not be addressed in class. Students should try to solve it at home and, whishing so, publish their solutions in the* ***#geral*** *Slack channel, so as to trigger discussion among other colleagues.*

Problem requirements:

In the context of a sporting competition (e.g. football league), suppose you have access to the history of all games of the competition, organized by date, in $db_1 :: [(Date, [Game])]$ *(using Haskell syntax). Also given is* $db_2 :: [(Game, [Player])]$ *indicating which palyers played in which game.*

A sport-tv commentator asks you to derive from db_1 *and from* db_2 *the list, ordered by player name, of the dates on which each player played, also ordered. Define, in Haskell, a function* f *implementing such a derivation:*

$$f :: [(Date, [Game])] \to [(Game, [Player])] \to [(Player, [Date])]$$

Challenged by these requirements, ChatGPT gave the solution given below in the black text boxes, which doesn't type but is the sort of solution to be expected.

In the context of this course, you can write ***far less*** *code to implement* f*!*

Why and how?

```
import Data.List (sort, nub)
type Date = String   -- You can replace String with an appropriate Date type
type Player = String
type Game = String
```

```
  -- Helper function to extract unique player names from a list of games
extractPlayers :: [(Game, [Player])] → [Player]
extractPlayers = nub · concatMap π2
  -- Helper function to map players to the dates they played on
mapPlayersToDates :: [(Date, [Game])] →
  [(Game, [Player])] → [(Player, [Date])]
mapPlayersToDates db1 db2 = [(player, sort $ nub playedDates)]
  where
    players = extractPlayers db2
    playedDates player = [date | (date, games) ← db1,
      any (λ(game, players) → player ∈ players ∧ game ∈ games) db2]
```

```
  -- Main function f
f :: [(Date, [Game])] → [(Game, [Player])] → [(Player, [Date])]
f db1 db2 = mapPlayersToDates db1 db2
```

```
  -- Example usage:
main :: IO ()
main = do
  let db1 = [("2023-10-01", ["Game1", "Game2"]),
    ("2023-10-02", ["Game2", "Game3"])]
  let db2 = [("Game1", ["PlayerA", "PlayerB"]),
    ("Game2", ["PlayerA", "PlayerC"]),
    ("Game3", ["PlayerB", "PlayerC"])]
  let result = f db1 db2
  print result
```

☐