Cálculo de Programas
Algebra of Programming

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

2025/26 - Ficha *(Exercise sheet)* nr. 7

1. O quadro abaixo representa a **propriedade universal** que define o combinador **catamorfismo** para listas finitas A^*, onde $\widehat{f}$ abrevia $\mathsf{uncurry} f$.

   *The table below depicts the **universal property** that defines the **catamorphism** combinator for finite lists A^*, where $\widehat{f}$ abbreviates $\mathsf{uncurry} f$:*

   Catamorfismo *(Catamorphism)*:

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

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

   Listas *(Lists)*:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

2. A função seguinte, em Haskell

The following function, in Haskell

$$\begin{aligned} &\mathit{sumprod}\ a\ [\,] = 0 \\ &\mathit{sumprod}\ a\ (h : t) = a * h + \mathit{sumprod}\ a\ t \end{aligned}$$

é o catamorfismo de listas

is the list-catamorphism

$$\mathit{sumprod}\ a \quad = \quad \llparenthesis\, [\mathsf{zero}\ , \mathsf{add} \cdot ((a*) \times id)]\, \rrparenthesis \qquad \text{(F1)}$$

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*

$$\mathit{sumprod}\ a \quad = \quad (a*) \cdot \mathsf{sum} \qquad \text{(F2)}$$

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

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

3. A igualdade que se segue

The following equality

$$f \cdot \mathsf{length} = \llparenthesis\, [\mathsf{zero}\ , (2+) \cdot \pi_2]\, \rrparenthesis$$

verifica-se para $f = (2*)$ ou $f = (2+)$? Use a lei de fusão-cata para justificar, por cálculo, a sua resposta.

holds for $f = (2*)$ *or* $f = (2+)$*? Use the cata-fusion law to justify, by calculation, your answer.*

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. O quadro abaixo representa a **propriedade universal** que define o combinador **catamorfismo** para números naturais $\mathbb{N}_0$.

The table below depicts the ***universal property*** *that defines the* ***catamorphism*** *combinator, for natural numbers* $\mathbb{N}_0$*.*

Catamorfismo *(Catamorphism)*:

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

Números naturais *(Natural numbers)*:

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

Recorde a lei de "fusão-cata"

Recall the "fusion-law"

$$f \cdot \llparenthesis\, g\, \rrparenthesis = \llparenthesis\, h\, \rrparenthesis \quad \Leftarrow \quad f \cdot g = h \cdot (id + f) \qquad \text{(F3)}$$

deduzida na aula teórica. Recorra a (F3) para demonstrar a propriedade:

proved in the theory class. Use (F3) to prove the property:

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

sabendo que $\mathsf{for}\ f\ i$ is $\llparenthesis\, [\underline{i}\ ,f]\,\rrparenthesis$.

knowing: $\mathsf{for}\ f\ i = \llparenthesis\, [\underline{i}\ ,f]\,\rrparenthesis$.

---

6. Mostre que as funções

Show that functions

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

são a mesma função. (Qual?)

are the same function. (Which one?)

---

7. Considere o catamorfismo $rep\ f = \llparenthesis\, [\underline{id}\ ,(f\cdot)]\,\rrparenthesis$. Comece por fazer um diagrama do catamorfismo e responda: Qual é o tipo de *rep*? O que faz *rep*?

Usando o combinador `cataNat g` da biblioteca `Nat.hs` para implementar $\llparenthesis\, g\,\rrparenthesis$, avalie no GHCi expressões como por exemplo $rep\ (2*)\ 0\ 3$, $rep\ (\texttt{"a"}{+\!\!+})\ 10\ \texttt{"b"}$ e veja se os resultados confirmam as suas respostas acima.

Consider catamorphism $rep\ f = \llparenthesis\, [\underline{id}\ ,(f\cdot)]\,\rrparenthesis$. *Draw a diagram of this catamorphism and answer: What is the type of rep? What does rep do?*

Using `cataNat g` *from library* `Nat.hs` *to implement* $\llparenthesis\, g\,\rrparenthesis$, *evaluate in GHCi expressions like* $rep\ (2*)\ 0\ 3$, $rep\ (\texttt{"a"}{+\!\!+})\ 10\ \texttt{"b"}$ *and check if the results confirm your answers above.*

---

8. 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:*

$$\llparenthesis\, g\,\rrparenthesis = g \cdot (id + \llparenthesis\, g\,\rrparenthesis) \cdot \mathsf{out} \qquad \text{(F4)}$$

De seguida implemente e teste a seguinte função:

Then implement and test de following function:

$$rep\ a = \llparenthesis\, [\mathsf{nil}\ ,(a{:})]\,\rrparenthesis \qquad \text{(F5)}$$

O que faz ela?

What is its purpose?

---

9. **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
```

□