Cálculo de Programas

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

2025/26 - Ficha nr.° 4

1. Demonstre a igualdade

$$[\underline{k}\ , \underline{k}] = \underline{k} \tag{F1}$$

recorrendo à propriedade universal-$+$ e a uma lei que qualquer função constante $\underline{k}$ satisfaz. (Consultar o formulário.)

---

2. Recorra às leis dos coprodutos para mostrar que a definição que conhece da função factorial,

$$\left\{ \begin{array}{l} fac\ 0 = 1 \\ fac\ (n+1) = (n+1) * fac\ n \end{array} \right. \tag{F2}$$

é equivalente à equação seguinte

$$fac \cdot [\underline{0}\ , \mathsf{succ}] = [\underline{1}\ , \mathsf{mul} \cdot \langle \mathsf{succ}, fac \rangle]$$

onde

$$\begin{array}{l} \mathsf{succ}\ n = n + 1 \\ \mathsf{mul}\ (a, b) = a * b \end{array}$$

---

3. A função $\mathsf{in} = [\underline{0}\ , \mathsf{succ}]$ da questão anterior exprime, para $\mathsf{succ}\ n = n+1$, a forma como os números naturais ($\mathbb{N}_0$) são gerados a partir do número 0, de acordo com o diagrama seguinte:

$$\begin{tikzcd} 1 \arrow[r, "i_1"] \arrow[dr, "\underline{0}"'] & 1 + \mathbb{N}_0 \arrow[d, "\mathsf{in}=[\underline{0}\ {,}\mathsf{succ}]"] & \mathbb{N}_0 \arrow[l, "i_2"'] \arrow[dl, "\mathsf{succ}"] \\ & \mathbb{N}_0 & \end{tikzcd} \tag{F3}$$

Sabendo que o tipo 1 coincide com o tipo $()$ em Haskell e que é habitado por um único elemento, também designado por $()$, calcule a inversa de in,

$$\left\{ \begin{array}{l} \mathsf{out}\ 0 = i_1\ () \\ \mathsf{out}\ (n+1) = i_2\ n \end{array} \right. \tag{F4}$$

resolvendo em ordem a out a equação

$$\mathsf{out} \cdot \mathsf{in} = id \tag{F5}$$

e introduzindo variáveis.

---

4. Verifique no GHCi que a seguinte função

$$fac = [\underline{1}\ , \mathsf{mul}] \cdot (id + \langle \mathsf{succ}, fac \rangle) \cdot \mathsf{out}$$

a que corresponde o diagrama

$$\begin{tikzcd} \mathbb{N}_0 \arrow[r, "\mathsf{out}"] \arrow[d, "fac"'] & 1 + \mathbb{N}_0 \arrow[d, "id + \langle \mathsf{succ}, fac \rangle"] \\ \mathbb{N}_0 & 1 + \mathbb{N}_0 \times \mathbb{N}_0 \arrow[l, "{[\underline{1}\ , \mathsf{mul}]}"] \end{tikzcd}$$

calcula o factorial da sua entrada (F1), assumindo out (F4) e mul $(a, b) = a * b$ já definidas.

---

5. Considere a seguinte declaração de um tipo de *árvores binárias*, em Haskell:

$$\mathbf{data}\ \mathsf{LTree}\ a = \mathit{Leaf}\ a \mid \mathit{Fork}\ (\mathsf{LTree}\ a, \mathsf{LTree}\ a)$$

Indagando os tipos dos construtores *Leaf* e *Fork*, por exemplo no GHCi,

```
*LTree> :t Fork
Fork :: (LTree a, LTree a) -> LTree a
*LTree> :t Leaf
Leaf :: a -> LTree a
```

faz sentido definir a função que mostra como construir árvores deste tipo:

$$\mathsf{in} = [\mathit{Leaf}\ , \mathit{Fork}] \tag{F6}$$

Desenhe um diagrama para esta função e calcule a sua inversa

$$\mathsf{out}\ (\mathit{Leaf}\ a) = i_1\ a$$
$$\mathsf{out}\ (\mathit{Fork}\ (x, y)) = i_2\ (x, y)$$

de novo resolvendo a equação $\mathsf{out} \cdot \mathsf{in} = id$ em ordem a out, agora para o (F6).

Finalmente, faça testes em Haskell que involvam a composição $\mathsf{in} \cdot \mathsf{out}$ e tire conclusões.

---

6. Considere o isomorfismo

$$\begin{tikzcd} (A + B) + C \arrow[rr, bend left, "\mathsf{coassocr}"] \arrow[rr, phantom, "\cong"] & & A + (B + C) \arrow[ll, bend left, "\mathsf{coassocl}"] \end{tikzcd}$$

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, a equação

$$\mathsf{coassocl} \cdot \underbrace{[id + i_1\ , i_2 \cdot i_2]}_{\mathsf{coassocr}} = id$$

Finalmente, exprima coassocl sob a forma de um programa em Haskell *não recorra* ao combinador "either"e teste as duas versões no GHCi.

---

7. Os isomorfismos

$$
\begin{tikzcd}
A \times (B + C) \arrow[rr, bend left, "\mathsf{distr}"] & \cong & A \times B + A \times C \arrow[ll, bend left, "\mathsf{undistr}"]
\end{tikzcd}
$$

estudados na aula teórica estão codificados na biblioteca $Cp.hs$. Supondo $A = String$, $B = \mathbb{B}$ e $C = \mathbb{Z}$, (a) aplique no GHCi undistr, alternativamente, aos pares $(\texttt{"CP"}, \textsc{True})$ ou $(\texttt{"LEI"}, 1)$; (b) verifique que $(\mathsf{distr} \cdot \mathsf{undistr})\ x = x$ para essas (e quaisquer outras) situações que possa testar.

---

8. A *lei da troca* (identifique-a no formulário) permite-nos exprimir determinadas funções sob duas formas alternativas, conforme desenhado no respectivo diagrama:

$$
[\langle f, g\rangle\ , \langle h, k\rangle] = \langle [f\ , h], [g\ , k]\rangle
\qquad
\begin{tikzcd}
A \arrow[r, "i_1"] \arrow[d, "f"'] \arrow[drr, "h"] & A + B & B \arrow[l, "i_2"'] \arrow[d, "k"] \arrow[dll, "g"] \\
C & C \times D \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] & D
\end{tikzcd}
\tag{F7}
$$

Demonstre esta lei recorrendo às propriedades (e.g. universais) dos produtos e dos coprodutos.

---

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.

Problem requirements:
Well-known services such as Google Maps, Google Analytics, YouTube, MapReduce etc. run on top of Bigtable or successors thereof. Such data systems rely on the so-called key-value NoSQL data model, which is widely adopted because of its efficiency and flexibility. Key-value stores can be regarded abstractly as lists of pairs $(K \times V)^*$ in which K is a datatype of keys *and V is a type of data* values*. Keys uniquely identify values. Key-value stores with the same type V of values can be glued together as the diagram suggests,*

$$
\begin{tikzcd}
((K + K') \times V)^* \arrow[rr, bend left, "unglue"] & & (K \times V)^* \times (K' \times V)^* \arrow[ll, bend left, "glue"]
\end{tikzcd}
$$

where $unglue$ performs the action opposite to $glue$.
*Define $glue$ and $unglue$ in Haskell structured along the functional combinators ($f \cdot g$, $\langle f, g\rangle$, $f \times g$ and so on) studied in this course and available from library Cp.hs. Use **diagrams** to plan your solutions, in which you should avoid re-inventing functions over lists already available from the Haskell standard libraries.*
$\square$