Cálculo de Programas
Algebra of Programming

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

2025/26 - Ficha nr.° 5

1. Deduza o tipo mais geral da função $\alpha = (id + \pi_1) \cdot i_2 \cdot \pi_2$ e represente-o através de um diagrama.

   Infer the most general type of function $\alpha = (id + \pi_1) \cdot i_2 \cdot \pi_2$ and draw it in a diagram of compositions.

---

2. Considere as seguintes funções elementares que respectivamente juntam ou duplicam informação:

   Let the following basic functions be given that, respectively, gather or duplicate information:

$$join = [id\ , id] \tag{F1}$$
$$dup = \langle id, id \rangle \tag{F2}$$

   Calcule (justificando) a propriedade grátis da função $\alpha = dup \cdot join$ e indique por que razão não pode calcular essa propriedade para $join \cdot dup$.

   Calculate (justifying) the free property of the function $\alpha = dup \cdot join$ and indicate why you cannot calculate this property for $join \cdot dup$.

---

3. Seja dada uma função $\triangledown$ da qual só sabe duas propriedades: $\triangledown \cdot i_1 = id$ e $\triangledown \cdot i_2 = id$. Mostre que, necessariamente, $\triangledown$ satisfaz também a propriedade natural

   Suppose that, about a function $\triangledown$, you only know two properties: $\triangledown \cdot i_1 = id$ and $\triangledown \cdot i_2 = id$. Show that, necessarily, $\triangledown$ also satisfies the natural property

$$f \cdot \triangledown = \triangledown \cdot (f + f) \tag{F3}$$

---

4. Seja dada uma função α cuja propriedade grátis é:

   Let α be a function with free property:

$$(f + h) \cdot \alpha \quad = \quad \alpha \cdot (f + g \times h) \tag{F4}$$

   Será esta propriedade suficiente para deduzir a definição de α? Justifique analiticamente.

   Can a definition of α be inferred from (F4)? Justify.

---

5. O formulário inclui as duas equivalências seguintes, válidas para qualquer isomorfismo α:

   Any isomorphism α satisfies the following equivalences (also given in the reference sheet),

$$\alpha \cdot g = h \equiv g = \alpha^\circ \cdot h \quad \text{(F5)}$$
$$g \cdot \alpha = h \equiv g = h \cdot \alpha^\circ \quad \text{(F6)}$$

Recorra a essas propriedades para mostrar que a igualdade

which can be useful to show that the equality

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

é equivalente à igualdade

is equivalent to:

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

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

*Prove this equivalence. (**Hint:** the free-property of* distr *shoudn't be ingored in the reasoning.)*

---

6. No cálculo de programas, as definições condicionais do tipo

Conditional expressions of pattern

$$h\ x = \textbf{if}\ p\ x\ \textbf{then}\ f\ x\ \textbf{else}\ g\ x \quad \text{(F7)}$$

são escritas usando o combinador ternário

are expressed in the algebra of programming by the ternary combinator

$$p \to f, g$$

conhecido pelo nome de *condicional de McCarthy*, cuja definição

known as the McCarthy conditionald*, whose definition*

$$p \to f, g \quad = \quad [f\ , g] \cdot p? \quad \text{(F8)}$$

vem no formulário. Baseie-se em leis desse formulário para demonstrar a chamada 2ª-lei de fusão do condicional:

can be found in reference sheet. Use this reference sheet to prove the so-called 2nd fusion-law of conditionals:

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

---

7. Numa máquina paralela pode fazer sentido, em (F7), não esperar por $p\ x$ para avaliar ou $f\ x$ ou $g\ x$, mas sim correr tudo em paralelo,

On a parallel machine it might make sense, concerning (F7), not to wait for $p\ x$ to evaluate either $f\ x$ or $g\ x$, but rather to run everything in parallel,

$$parallel\ p\ f\ g = \langle \langle f, g \rangle, p \rangle$$

e depois fazer a escolha do resultado:

and then choose the outcome:

$$choose = \pi_2 \to \pi_1 \cdot \pi_1, \pi_2 \cdot \pi_1$$

Mostre que, de facto:

Show that, indeed:

$$choose \cdot parallel\ p\ f\ g \quad = \quad p \rightarrow f, g$$

---

8. Sabendo que as igualdades

Assuming

$$p \rightarrow k, k \ = \ k \tag{F9}$$
$$(p? + p?) \cdot p? \ = \ (i_1 + i_2) \cdot p? \tag{F10}$$

se verificam, demonstre as seguintes propriedades do mesmo combinador:

prove the following laws of the McCarthy conditional:

$$\langle (p \rightarrow f, h), (p \rightarrow g, i) \rangle \ = \ p \rightarrow \langle f, g \rangle, \langle h, i \rangle \tag{F11}$$
$$\langle f, (p \rightarrow g, h) \rangle \ = \ p \rightarrow \langle f, g \rangle, \langle f, h \rangle \tag{F12}$$
$$p \rightarrow (p \rightarrow a, b), (p \rightarrow c, d) \ = \ p \rightarrow a, d \tag{F13}$$

---

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.
Dão-se a seguir os requisitos do problema.

***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.*
The requirements of the problem are given below.

***Problem requirements**: The solution given for a previous problem,*

$$store\ c = \mathsf{take}\ 10 \cdot nub \cdot (c{:}) \tag{F14}$$

calls the standard function

$nub :: (Eq\ a) \Rightarrow [\,a\,] \rightarrow [\,a\,]$

available from the Data.List library in Haskell.
After inspecting the standard implementation of this function, define f so that

$nub = [\mathsf{nil}\ , \mathsf{cons}] \cdot f.$

is an alternative to the standard definition, where $\mathsf{nil}\ \_ = [\,]$ *and* $\mathsf{cons}\ (h, t) = h : t$*. Check that* $store\ c$ *(F14) works properly once the standard* nub *is replaced by yours.*

***Important**: Structure your solution across the* $f \cdot g$, $\langle f, g \rangle$, $f \times g$, $[f\ , g]$ *and* $f + g$ *combinators available from library Cp.hs. Use **diagrams** to plan your solution, in which you should avoid re-inventing functions over lists already available in the Haskell standard libraries.*
□