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.° 6

1. Mostre que a propriedade de cancelamento da exponenciação

Show that the cancellation property

$$\mathsf{ap} \cdot (\overline{f} \times id) = f \tag{F1}$$

corresponde à definição

is nothing but the definition

$$\mathsf{curry}\ f\ \ a\ \ b = f\ \ (a, b)$$

quando se escreve $\mathsf{curry}\ f$ em lugar de $\overline{f}$.

once $\mathsf{curry}\ f$ *is written instead of* $\overline{f}$.

---

2. Mostre que a definição *pointwise* de $\mathsf{uncurry}$ se pode obter também de (F1) fazendo $f := \mathsf{uncurry}\ g$, introduzindo variáveis e simplificando.

Show that the pointwise definition of $\mathsf{uncurry}$ *can also be obtained from (F1) by instantiating* $f := \mathsf{uncurry}\ g$*, introducing variables and simplifying.*

---

3. Prove a igualdade

Prove the equality

$$\overline{f \cdot (g \times h)} \ = \ \overline{\mathsf{ap} \cdot (id \times h)} \cdot \overline{f} \cdot g \tag{F2}$$

usando as leis das exponenciais e dos produtos.

using the laws of products and exponentials.

---

4. É dada a definição

Let flip *be defined by*

$$\mathsf{flip}\ f = \overline{\widehat{f} \cdot \mathsf{swap}} \tag{F3}$$

de acordo com:

according to:

$$\begin{array}{ccccccc} (C^B)^A & \cong & C^{A \times B} & \cong & C^{B \times A} & \cong & (C^A)^B \\ f & \mapsto & \widehat{f} & \mapsto & \widehat{f}.\mathsf{swap} & \mapsto & \overline{\widehat{f} \cdot \mathsf{swap}} = \mathsf{flip}\ f \end{array}$$

Mostre que flip é um isomorfismo por ser a sua própria inversa:

Show that it is an isomorphism because it is its own inverse:

$$\mathsf{flip}\ (\mathsf{flip}\ f) = f \tag{F4}$$

Mostre ainda que:

Furthermore show:

$$\mathsf{flip}\ f\ x\ y = f\ y\ x$$

---

5. Mostre que

Show that

$$junc \cdot unjunc = id \tag{F5}$$
$$unjunc \cdot junc = id \tag{F6}$$

se verificam, onde

hold for

$$\begin{tikzcd} A^{B+C} \arrow[rr, bend left, "unjunc"] \arrow[rr, phantom, "\cong"] & & A^B \times A^C \arrow[ll, bend left, "junc"] \end{tikzcd} \qquad \begin{cases} junc\ (f, g) = [f\ , g] \\ unjunc\ k = (k \cdot i_1, k \cdot i_2) \end{cases} \tag{F7}$$

---

6. O código que se segue, escrito em Haskell, implementa a noção de ciclo-for, onde b é o corpo (“body”) do ciclo e i é a sua inicialização:

The following Haskell code implements a for *-loop where b is the loop-body and i is its initialization:*

$$\begin{cases} \mathsf{for}\ b\ i\ 0 = i \\ \mathsf{for}\ b\ i\ (n+1) = b\ (\mathsf{for}\ b\ i\ n) \end{cases} \tag{F8}$$

Tem-se pois o tipo:

We thus have type:

$$\mathsf{for} : (a \to a) \to a \to \mathbb{N}_0 \to a.$$

Mostre que for $b\ i$ satisfaz o diagrama que se segue, para um dado g:

Show that for $b\ i$ *fits in the following diagram, for some g:*

$$\begin{tikzcd} \mathbb{N}_0 \arrow[rr, bend left, "\mathsf{out}"] \arrow[rr, phantom, "\cong"] \arrow[d, "\mathsf{for}\ b\ i"'] & & 1 + \mathbb{N}_0 \arrow[ll, bend left, "\mathsf{in}"] \arrow[d, "id + \mathsf{for}\ b\ i"] \\ A & & 1 + A \arrow[ll, bend left, "g"] \end{tikzcd}$$

onde

where

$$\begin{cases} \mathsf{in} = [\mathsf{zero}\ , \mathsf{succ}] \\ \mathsf{zero}\ \_ = 0 \\ \mathsf{succ}\ n = n + 1 \end{cases} \tag{F9}$$

Sugestão: faça $g = [g_1\ , g_2]$ e resolva o problema em ordem a g_1 e g_2.

***Hint**: make $g = [g_1\ , g_2]$ and solve the problem for g_1 and g_2.*

---

7. Na sequência da questão anterior, codifique (F8) em Haskell bem como

As follow up of the previous question, encode (F8) in Haskell as well as

$$f = \pi_2 \cdot aux \ \mathbf{where}\ aux = \mathsf{for}\ \langle \mathsf{succ} \cdot \pi_1, \mathsf{mul} \rangle\ (1,1) \qquad \text{(F10)}$$

e inspecione o seu comportamento. Que função f é essa?

and inspect its behavior. Which function is f?

---

8. Mostre que $(a+)$ dada a seguir é um ciclo $\mathsf{for}\ b\ i$ (F8) para um dado b e um dado i — descubra quais:

Show that $(a+)$ given next is a for-loop $\mathsf{for}\ b\ i$ *(F8) for b and i to be calculated:*

$$\left\{ \begin{array}{l} a + 0 = a \\ a + (n+1) = 1 + (a+n) \end{array} \right. \qquad \text{(F11)}$$

---

9. Qualquer função $k = \mathsf{for}\ f\ i$ pode ser codificada em sintaxe C escrevendo

Any function $k = \mathsf{for}\ f\ i$ can be encoded in the syntax of C by writing:

```
int k(int n) {
  int r=i;
  int j;
  for (j=1;j<n+1;j++) {r=f(r);}
  return r;
};
```

Escreva em sintaxe C as funções $(a*) = \mathsf{for}\ (a+)\ 0$ e outros catamorfismos de naturais de que se tenha falado nas aulas da UC.

Encode function $(a*) = \mathsf{for}\ (a+)\ 0$ in C and other catamorphisms that have been discussed in the previous classes.

---

10. **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 following function*

$$func :: Eq\ a \Rightarrow b \rightarrow [(a,b)] \rightarrow (a \rightarrow b)$$
$$func\ b = (maybe\ b\ id \cdot) \cdot \mathsf{flip}\ lookup$$

"functionalizes" *a finite list of $(key, value)$ pairs by converting it to a function from keys to values. The first parameter provides a default value for keys that cannot be found in the list.*
As example, let us have a list of people (where the key is some numeric id),

$$a = [(140999000, \texttt{"Manuel"}), (200100300, \texttt{"Mary"}), (000111222, \texttt{"Teresa"})]$$

their nationalities (if known),

$$b = [(140999000, \texttt{"PT"}), (200100300, \texttt{"UK"})]$$

and places of residence (if known):

$$c = [(140999000, \texttt{"Braga"}), (200100300, \texttt{"Porto"}), (151999000, \texttt{"Lisbon"})]$$

Using only func, $\langle f, g\rangle$, π_1, π_2, map *and nub, write a Haskell expression representing the following data aggregation:*

Id	*Name*	*Country*	*Residence*
140999000	$Manuel$	PT	$Braga$
200100300	$Mary$	UK	$Porto$
000111222	$Teresa$	*?*	-
151999000	$(Unknown)$	*?*	$Lisbon$

□