Cálculo de Programas

Resolução - Ficha 01

Eduardo Freitas Fernandes

2026

Exercício 1

$$
\begin{aligned}
&\mathit{length}\ [\,] = 0\\
&\mathit{length}\ (x : xs) = 1 + \mathit{length}\ xs\\
&\mathit{reverse}\ [\,] = [\,]\\
&\mathit{reverse}\ (x : xs) = \mathit{reverse}\ xs +\!\!+ [x]
\end{aligned}
$$

Exercício 2

$$\mathit{take}\ m\ (\mathit{take}\ n\ x) = \mathit{take}\ (\mathit{min}\ m\ n)\ x$$

Exercício 3

$$
\begin{aligned}
&\mathsf{map}\ \_\ [\,] = [\,]\\
&\mathsf{map}\ f\ (x : xs) = f\ x : \mathsf{map}\ f\ xs\\
&\mathit{filter}\ \_\ [\,] = [\,]\\
&\mathit{filter}\ f\ (x : xs) = \textbf{if}\ f\ x\ \textbf{then}\ x : \mathit{filter}\ f\ xs\ \textbf{else}\ \mathit{filter}\ f\ xs\\
&\widehat{f}\ (x, y) = f\ x\ y\\
&\overline{f}\ x\ y = f\ (x, y)\\
&\mathit{flip}\ f\ x\ y = f\ y\ x
\end{aligned}
$$

Exercício 4

$$(f \cdot g)\ x = f\ (g\ x) = 2 * (x + 1) = 2 * x + 2$$

$$(f \cdot g)\ x = f\ (g\ x) = \mathsf{succ}\ (2 * x) = 2 * x + 1$$

$$(f \cdot g)\ x = f\ (g\ x) = \mathsf{succ}\ (\mathit{length}\ x) = \mathit{length}\ x + 1$$

$$(f \cdot g)\ (x, y) = f\ (g\ (x, y)) = (\mathsf{succ} \cdot (2*))\ (x + y) = \mathsf{succ}\ (2 * (x + y)) = 2 * x + 2 * y + 1$$

Exercício 5

$$
\begin{aligned}
&(f \cdot g) \cdot h = f \cdot (g \cdot h)\\
\equiv\ &\quad \{\ \mathsf{pointwise}\ \}\\
&((f \cdot g) \cdot h)\ x = (f \cdot (g \cdot h))\ x\\
\equiv\ &\quad \{\ \mathsf{Def.\ comp}\ \}
\end{aligned}
$$

$$
\begin{aligned}
& (f \cdot g)\ (h\ x) = f\ ((g \cdot h)\ x) \\
\equiv & \quad \{\ \text{Def. comp}\ \} \\
& f\ (g\ (h\ x)) = f\ (g\ (h\ x))
\end{aligned}
$$

Exercício 6

$$
\begin{cases}
(f \cdot id)\ x = f\ (id\ x) = f\ x \\
(id \cdot f)\ x = id\ (f\ x) = f\ x
\end{cases}
$$

Exercício 7

alínea a)

$$
\begin{aligned}
& store\ 7\ [1\,..\,10] \\
\equiv & \quad \{\ (F2)\ \} \\
& (take\ 10 \cdot nub \cdot (7:))\ [1\,..\,10] \\
\equiv & \quad \{\ (F1)\ \} \\
& take\ 10\ (nub\ (7 : [1\,..\,10])) \\
\equiv & \quad \{\ \text{Def. } (7:)\ \} \\
& take\ 10\ (nub\ [7,1,2,3,4,5,6,7,8,9,10]) \\
\equiv & \quad \{\ \text{Def. } nub\ \} \\
& take\ 10\ [7,1,2,3,4,5,6,8,9,10] \\
\equiv & \quad \{\ \text{Def. } take\ \} \\
& [7,1,2,3,4,5,6,8,9,10]
\end{aligned}
$$

alínea b) o requisito (a) é cumprido mas os restantes não, pois a primeira operação a ser feita é a remoção de duplicados. Esta operação deve ser feita após adicionar *c*. O requisito (b) não é cumprido, pois primeiro seleciona-se os 10 primeiros elementos e depois adiciona-se *c*, logo a lista final terá no máximo 11 elementos, violando o requisito (c).

alínea c) o requisito (c) é violado, pois primeiro retira-se os primeiros 10 elementos e adiciona-se *c*, logo a lista final terá no máximo 11 elementos (caso a função *nub* não efetue mudanças na lista).

Exercício 8 O resultado será `["Mary","Manuel","Tia Irene","Augusto"]`. Neste exemplo é evidenciado o facto das funções en Haskell só receberem um argumento. Ao analisar a composição de funções em (F2):

- $(c:) :: [a] \to [a]$
- $nub :: [a] \to [a]$
- $take\ 10 :: [a] \to [a]$

verificamos que a função *store* não recebe dois argumentos, recebe um argumento *c* e devolve uma função do tipo $store\ c :: [a] \to [a]$. Isto acontece porque Haskell aplica **currying** automaticamente.