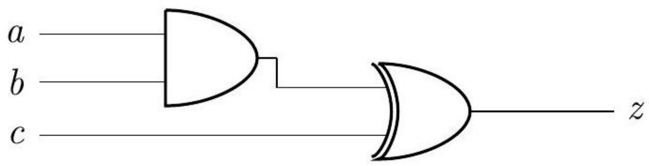
CP - Ficha 2

Exercício 1

> O circuito booleano
>
> 
>
>
> pode descrever-se pela função f que se segue,
>
> $$\begin{cases} f : (\mathbb{B} \times \mathbb{B}) \times \mathbb{B} \to \mathbb{B} \\ f = xor \cdot (and \times id) \end{cases} \tag{F1}$$
>
> onde $and\ (a, b) = a \wedge b$ e $xor\ (x, y) = x \oplus y$.
>
> a) Mostre que f se pode também definir como se segue:
>
> $$f\ ((a, b), c) = (a \wedge b) \oplus c$$
>
> b) Qual o tipo da função $g = \langle \pi_1, f \rangle$?

Resolução 1

a)

$$\begin{aligned}
f\ ((a,b),c) &= (a \wedge b) \oplus c \\
&= xor\ ((a \wedge b), c) && \text{(Def. xor)} \\
&= xor\ (and\ (a,b), c) && \text{(Def. and)} \\
&= xor\ (and\ (a,b), id\ c) && \text{(74: Def-id)} \\
&= xor\ ((and \times id)\ ((a,b),c)) && \text{(78: Def-}\times\text{)} \\
&= xor \cdot (and \times id)\ ((a,b),c) && \text{(73: Def-comp)} \\
&\equiv && \text{(72: Ig. Ext.)} \\
f &= xor \cdot (and \times id)
\end{aligned}$$

b)

$$f : (\mathbb{B} \times \mathbb{B}) \times \mathbb{B} \to \mathbb{B}$$
$$\pi_1 : A \times C \to A$$

$$\begin{array}{rl} \text{Se} & A = \mathbb{B} \times \mathbb{B} \\ \text{e} & C = \mathbb{B} \\ \text{então} & \pi_1 : (\mathbb{B} \times \mathbb{B}) \times \mathbb{B} \to \mathbb{B} \times \mathbb{B} \end{array}$$

$$\begin{array}{rl} \text{Como} & g = \langle \pi_1, f \rangle \\ \text{então} & g : (\mathbb{B} \times \mathbb{B}) \times \mathbb{B} \to (\mathbb{B} \times \mathbb{B}) \times \mathbb{B} \end{array}$$

$$\begin{tikzcd}
& (\mathbb{B} \times \mathbb{B}) \times \mathbb{B} \arrow[dl, "\pi_1"'] \arrow[d, "g"] \arrow[dr, "f"] & \\
\mathbb{B} \times \mathbb{B} & (\mathbb{B} \times \mathbb{B}) \times \mathbb{B} \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] & \mathbb{B}
\end{tikzcd}$$

Exercício 2

> Implemente e teste f (F1) no GHCi, após carregar a biblioteca $Cp.hs$ disponível no material pedagógico.
>
> NB: recorde que $x \oplus y = x \neq y$.

Resolução 2

```
ghci> :load lib/Cp.hs
[1 of 1] Compiling Cp               ( lib/Cp.hs, interpreted )
Ok, one module loaded.
ghci> f ((a, b), c) = (a && b) /= c
ghci> :type f
f :: ((Bool, Bool), Bool) -> Bool
```

Tabela de verdade:

a	b	c	$f\ ((a,\ b),\ c)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Exercício 3

> Defina no GHCi o seguinte tipo de dados:
>
> ```haskell
> data X = B Bool | P (Bool,Int)
> ```
>
> Peça ao GHCi informação sobre os tipos de B e de P e deduza que são funções tais que $f = [B, P]$ faz sentido. Qual é o tipo de f?
>
> NB: em Haskell a alternativa [f, g] escreve-se `either f g`.

Resolução 3

```
ghci> data X = B Bool | P (Bool, Int)
ghci> :type B
B :: Bool -> X
ghci> :type P
P :: (Bool, Int) -> X
ghci> :type either B P
either B P :: Either Bool (Bool, Int) -> X
```

Como $f = [B, P] = \text{either}\ B\ P$, então f é uma função que recebe
um valor do tipo `Either B P`, ou seja `Either Bool (Bool, Int)`,
e devolve um valor do tipo X.

```haskell
f :: Either Bool (Bool, Int) -> X
```

Equivalente a:

$$f\ :\ \mathbb{B} + (\mathbb{B} \times \text{Int}) \to \text{X}$$

$$\begin{tikzcd}
\mathbb{B} \arrow[r, "i_1"] \arrow[dr, "B"] & \mathbb{B} + (\mathbb{B} \times \mathbf{Int}) \arrow[d, "{[B,P]}"] & \mathbb{B} \times \mathbf{Int} \arrow[l, "i_2"'] \arrow[dl, "P"'] \\
 & \mathbf{X} &
\end{tikzcd}$$

Exercício 4

> O combinador $\langle f, g\rangle$ - isto é, *"f em paralelo com g"* - satisfaz a seguinte propriedade, dita **universal**:
>
> $$k = \langle f, g\rangle \quad \equiv \quad \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases} \tag{F2}$$
>
> Identifique-a no formulário. Que outra propriedade desse formulário obtém fazendo $k = id$ e simplificando?

Resolução 4

$$k = \langle f, g\rangle \quad \Leftrightarrow \quad \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases} \tag{6: Universal-$\times$}$$

$$\begin{aligned}
\text{Se} \quad & k = id \\
\text{então} \quad & id = \langle f, g\rangle \Leftrightarrow \begin{cases} \pi_1 \cdot id = f \\ \pi_2 \cdot id = g \end{cases} \\
& \equiv \qquad\qquad\qquad\qquad\qquad\qquad (1\text{: Natural-id}) \\
& id = \langle f, g\rangle \Leftrightarrow \begin{cases} \pi_1 = f \\ \pi_2 = g \end{cases} \\
\text{Substituindo temos:} \quad & id = \langle \pi_1, \pi_2\rangle \\
& \equiv \\
& \langle \pi_1, \pi_2\rangle = id_{A\times B} \qquad\qquad (8\text{: Reflexão-}\times)
\end{aligned}$$

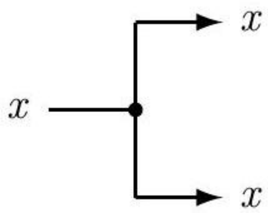
Exercício 5

> Derive a partir de (F2) a lei
>
> $$\langle h,\ k\rangle \cdot f = \langle h \cdot f,\ k \cdot f\rangle$$
>
> que também consta desse formulário sob a designação $\text{fusão-}\times$.

Resolução 5

$$\begin{array}{l}
\langle h,\ k\rangle \cdot f = \langle h \cdot f,\ k \cdot f\rangle \\
\equiv \qquad \text{(6: Universal-}\times\text{, 2: Assoc-comp)} \\
\begin{cases} (\pi_1 \cdot \langle h,\ k\rangle) \cdot f = h \cdot f \\ (\pi_2 \cdot \langle h,\ k\rangle) \cdot f = k \cdot f \end{cases} \\
\equiv \qquad \text{(7: Cancelamento-}\times\text{)} \\
\begin{cases} h \cdot f = h \cdot f \\ k \cdot f = k \cdot f \end{cases} \quad \text{c.q.m.}
\end{array}$$

Exercício 6

> Uma das operações essenciais em processamento da informação é a sua *duplicação*:
>
> 
>
>
> $$dup\ x = (x, x)$$
>
> Recorra à lei de fusão-$\times$ para demonstrar a seguinte propriedade da duplicação de informação:
>
> $$dup \cdot f = \langle f, f \rangle$$

Resolução 6

$$\begin{array}{ll}
& dup\ x = (x, x) \\
\equiv & \hfill (74\text{: Def-id}) \\
& dup\ x = (id\ x, id\ x) \\
\equiv & \hfill (77\text{: Def-split}) \\
& dup\ x = \langle id, id \rangle\ x \\
\equiv & \hfill (72\text{: Ig. Ext.}) \\
& dup = \langle id, id \rangle
\end{array}$$

$$\begin{array}{lll}
\text{Então} & & dup \cdot f = \langle id, id \rangle \cdot f \\
& \equiv & \hfill (9\text{: Fusão-}\times) \\
& & dup \cdot f = \langle id \cdot f, id \cdot f \rangle \\
& \equiv & \hfill (1\text{: Natural-id}) \\
& & dup \cdot f = \langle f, f \rangle \quad \text{c.q.m.}
\end{array}$$

Exercício 7

> Considere a função:
>
> $$\mathrm{xr} = \langle \pi_1 \times id,\ \pi_2 \cdot \pi_1 \rangle$$
>
> Mostre que xr satisfaz a propriedade
>
> $$\mathrm{xr} \cdot \langle \langle f, g \rangle, h \rangle = \langle \langle f, h \rangle, g \rangle$$
>
> para todo f, g e h.

Resolução 7

$$
\begin{aligned}
& \langle \pi_1 \times id,\ \pi_2 \cdot \pi_1 \rangle \cdot \langle \langle f, g \rangle, h \rangle \\
& = \qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(9: Fusão-}\times\text{)} \\
& \langle (\pi_1 \times id) \cdot \langle \langle f, g \rangle, h \rangle,\ (\pi_2 \cdot \pi_1) \cdot \langle \langle f, g \rangle, h \rangle \rangle \\
& = \qquad\qquad\qquad\qquad \text{(11: Absorção-}\times\text{, 2: Assoc-comp)} \\
& \langle \langle \pi_1 \cdot \langle f, g \rangle,\ id \cdot h \rangle,\ \pi_2 \cdot (\pi_1 \cdot \langle \langle f, g \rangle, h \rangle) \rangle \\
& = \qquad\qquad\qquad\qquad \text{(7: Cancelamento-}\times\text{, 1: Natural-id)} \\
& \langle \langle f,\ h \rangle,\ g \rangle \quad \text{c.q.m.}
\end{aligned}
$$

Extra: Qual o tipo de xr?

$$
\begin{tikzcd}
& (A \times B) \times C \arrow[dl, "\pi_1 \times id"'] \arrow[d, "\mathrm{xr}"] \arrow[dr, "\pi_2 \cdot \pi_1"] & \\
A \times C & (A \times C) \times B \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] & B
\end{tikzcd}
$$

Exercício 8

> O combinador
>
> ```haskell
> const :: a -> b -> a
> const a b = a
> ```
>
> está disponível em Haskell para construir funções constantes, sendo habitual designarmos $\text{const } k$ por $\underline{k}$. Demonstre a igualdade
>
> $$\underline{(b, a)} = \langle \underline{b}, \underline{a} \rangle$$
>
> a partir da propriedade universal do produto e das propriedades das funções constantes que constam do formulário.

Resolução 8

$$
\begin{array}{ll}
\underline{(b, a)} = \langle \underline{b}, \underline{a} \rangle & \\
\equiv & \{ \text{(6: Universal-}\times) \} \\
\underline{(b, a)} = \begin{cases} \pi_1 \cdot \underline{(b, a)} = \underline{b} \\ \pi_2 \cdot \underline{(b, a)} = \underline{a} \end{cases} & \\
\equiv & \{ \text{(4: Absorção-const)} \} \\
\begin{cases} \pi_1 \cdot \underline{(b, a)} = \underline{b} \\ \pi_2 \cdot \underline{(b, a)} = \underline{a} \end{cases} & \\
\equiv & \{ \text{(79: Def-proj)} \} \\
\begin{cases} \underline{b} = \underline{b} \\ \underline{a} = \underline{a} \end{cases} \quad \text{c.q.m.} &
\end{array}
$$

Exercício 9

> Determine o tipo da função α que se segue:
>
> $$\alpha = [\langle \underline{\mathrm{False}}, id \rangle, \langle \underline{\mathrm{True}}, id \rangle]$$

Resolução 9

Para definir o tipo mais geral de α, vamos primeiro determinar o tipo mais geral de cada uma das funções que a compõem:

$$\begin{gathered} \underline{False} : A \to \mathrm{Bool} \\ \langle \underline{False}, id \rangle : \mathrm{A} \to \mathrm{Bool} \times \mathrm{A} \end{gathered}$$

$$\begin{gathered} \underline{True} : B \to \mathrm{Bool} \\ \langle \underline{True}, id \rangle : \mathrm{B} \to \mathrm{Bool} \times \mathrm{B} \end{gathered}$$

No entanto a composição alternativa condiciona o tipo de saída dos seus argumentos.

Temos:

$$\begin{gathered} \underline{True} : A \to \mathrm{Bool} \\ \langle \underline{True}, id \rangle : \mathrm{A} \to \mathrm{Bool} \times \mathrm{A} \end{gathered}$$

Logo o tipo de α é:

$$\alpha : A + A \to \mathrm{Bool} \times A$$

$$\begin{tikzcd}
A \arrow[r, "i_1"] \arrow[dr, "\langle \underline{False}, id \rangle"'] & A + A \arrow[d, "\alpha"] & A \arrow[l, "i_2"'] \arrow[dl, "\langle \underline{True}, id \rangle"] \\
& \mathrm{Bool} \times A &
\end{tikzcd}$$

Exercício 10

> **Questão Prática**: - […] Dão-se a seguir os requisitos do problema.
>
> > ***Problem requirements**: Given a name, for instance `"Jose Nuno Oliveira"` we wish to obtain its acronym and its short version, as suggested below:*
> >
> > ```
> > *Cp> acronym "Jose Nuno Oliveira"
> > "JNO"
> > *Cp> short "Jose Nuno Oliveira"
> > "Jose Oliveira"
> > *Cp>
> > ```
> >
> > *Define*
> >
> > ```
> > acronym = ...
> > short = ...
> > ```
> >
> > *subject to the following restrictions:*
> > *• you cannot use argument variables ($x, y, \ldots$)*
> > *• you can use function composition $f \cdot g$ and the parallel combinator $\langle f, g \rangle$ as well as any function available from module $Cp.hs$*
> > *• you can resort to Haskell standard functions such as e.g. `map` , `filter` and so on.*

Resolução 10

```haskell
acronym = map head . words
short = uncurry (++) . split head ((" " ++) . last) . words
```