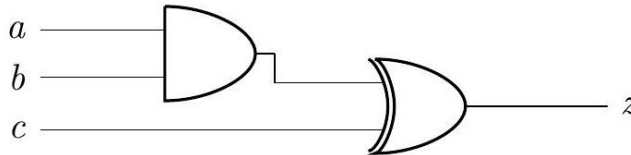


CP - Ficha 2

Exercício 1

O circuito booleano



pode descrever-se pela função f que se segue,

$$\begin{cases} f : (\mathbb{B} \times \mathbb{B}) \times \mathbb{B} \rightarrow \mathbb{B} \\ f = \text{xor} \cdot (\text{and} \times \text{id}) \end{cases} \quad (\text{F1})$$

onde $\text{and}(a, b) = a \wedge b$ e $\text{xor}(x, y) = x \oplus y$.

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)

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

b)

$$f : (\mathbb{B} \times \mathbb{B}) \times \mathbb{B} \rightarrow \mathbb{B}$$

$$\pi_1 : A \times C \rightarrow A$$

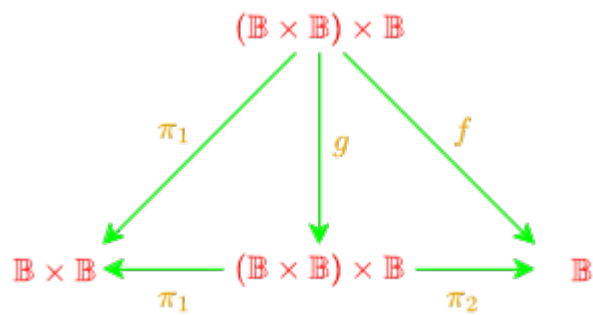
Se $A = \mathbb{B} \times \mathbb{B}$

e $C = \mathbb{B}$

então $\pi_1 : (\mathbb{B} \times \mathbb{B}) \times \mathbb{B} \rightarrow \mathbb{B} \times \mathbb{B}$

Como $g = \langle \pi_1, f \rangle$

então $g : (\mathbb{B} \times \mathbb{B}) \times \mathbb{B} \rightarrow (\mathbb{B} \times \mathbb{B}) \times \mathbb{B}$



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:

```
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

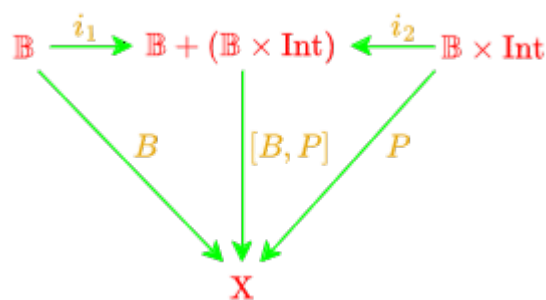
```
ghci> data X = B Bool | P (Bool, Int)
ghci> :type B
B :: Bool -> X
ghci> :type P
P :: (Bool, Int) -> X
ghci> :type either B P
either B P :: Either Bool (Bool, Int) -> X
```

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 .

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

Equivalente a:

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



Exercício 4

O combinador $\langle f, g \rangle$ - isto é, "*f em paralelo com g*" - satisfaz a seguinte propriedade, dita **universal**:

$$k = \langle f, g \rangle \quad \equiv \quad \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases} \quad (\text{F2})$$

Identifique-a no [formulário](#). Que outra propriedade desse formulário obtém fazendo $k = id$ e simplificando?

Resolução 4

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

Se $k = id$

$$\text{então } id = \langle f, g \rangle \Leftrightarrow \begin{cases} \pi_1 \cdot id = f \\ \pi_2 \cdot id = g \end{cases}$$

\equiv

(1: Natural-id)

$$id = \langle f, g \rangle \Leftrightarrow \begin{cases} \pi_1 = f \\ \pi_2 = g \end{cases}$$

Substituindo temos: $id = \langle \pi_1, \pi_2 \rangle$

\equiv

$$\langle \pi_1, \pi_2 \rangle = id_{A \times B} \quad (8: \text{Reflexão-}\times)$$

Exercício 5

Derive a partir de (F2) a lei

$$\langle h, k \rangle \cdot f = \langle h \cdot f, k \cdot f \rangle$$

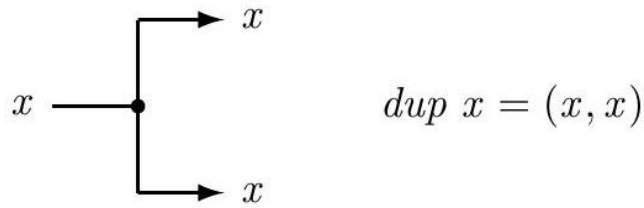
que também consta desse [formulário](#) sob a designação fusão- \times .

Resolução 5

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

Exercício 6

Uma das operações essenciais em processamento da informação é a sua *duplicação*:



Recorra à lei de fusão- \times para demonstrar a seguinte propriedade da duplicação de informação:

$$dup \cdot f = \langle f, f \rangle$$

Resolução 6

$$dup\ x = (x, x)$$

\equiv

(74: Def-id)

$$dup\ x = (id\ x, id\ x)$$

\equiv

(77: Def-split)

$$dup\ x = \langle id, id \rangle x$$

\equiv

(72: Ig. Ext.)

$$dup = \langle id, id \rangle$$

Então $dup \cdot f = \langle id, id \rangle \cdot f$

\equiv

(9: Fusão- \times)

$$dup \cdot f = \langle id \cdot f, id \cdot f \rangle$$

\equiv

(1: Natural-id)

$$dup \cdot f = \langle f, f \rangle \quad \text{c.q.m.}$$

Exercício 7

Considere a função:

$$\mathbf{xr} = \langle \pi_1 \times id, \pi_2 \cdot \pi_1 \rangle$$

Mostre que \mathbf{xr} satisfaz a propriedade

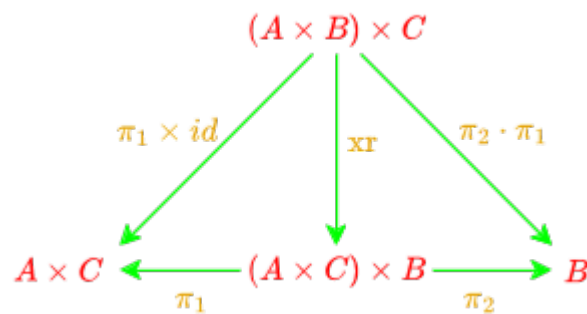
$$\mathbf{xr} \cdot \langle \langle f, g \rangle, h \rangle = \langle \langle f, h \rangle, g \rangle$$

para todo f, g e h .

Resolução 7

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

Extra: Qual o tipo de \mathbf{xr} ?



Exercício 8

O combinador

const :: a -> b -> a

const a b = a

está disponível em Haskell para construir funções constantes, sendo habitual designarmos `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

$$\underline{(b, a)} = \langle \underline{b}, \underline{a} \rangle$$

\equiv

(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

(4: Absorção-const)

$$\begin{cases} \pi_1 \cdot \underline{(b, a)} = \underline{b} \\ \pi_2 \cdot \underline{(b, a)} = \underline{a} \end{cases}$$

\equiv

(79: Def-proj)

$$\begin{cases} \underline{b} = \underline{b} \\ \underline{a} = \underline{a} \end{cases} \quad \text{c.q.m.}$$

Exercício 9

Determine o tipo da função α que se segue:

$$\alpha = [\langle \underline{\text{False}}, id \rangle, \langle \underline{\text{True}}, id \rangle]$$

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:

$$\underline{\text{False}} : A \rightarrow \text{Bool}$$

$$\langle \underline{\text{False}}, id \rangle : A \rightarrow \text{Bool} \times A$$

$$\underline{\text{True}} : B \rightarrow \text{Bool}$$

$$\langle \underline{\text{True}}, id \rangle : B \rightarrow \text{Bool} \times B$$

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

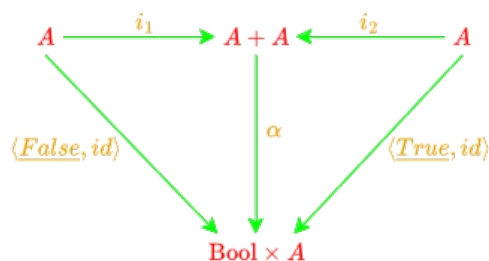
Temos:

$$\underline{\text{True}} : A \rightarrow \text{Bool}$$

$$\langle \underline{\text{True}}, id \rangle : A \rightarrow \text{Bool} \times A$$

Logo o tipo de α é:

$$\alpha : A + A \rightarrow \text{Bool} \times A$$



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, \dots)
- 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

acronym = map head . words

short = uncurry (++) . split head ((" " ++) . last) . words

