Cálculo de Programas

2.° ano
Lic. Ciências da Computação e Mestrado Integrado em Engenharia Informática
UNIVERSIDADE DO MINHO

2020/21 - Ficha nr.° 5

1. Considere a função iso $= \langle ! + !, [id\ , id] \rangle$, onde $! : A \to 1$ designa a única função constante que habita o tipo $A \to 1$.[1]

   (a) Identifique o isomorfismo que ela testemunha, desenhando-a sob a forma de um diagrama de tipos.

   (b) Derive a partir desse diagrama a propriedade (dita *grátis*) de iso,

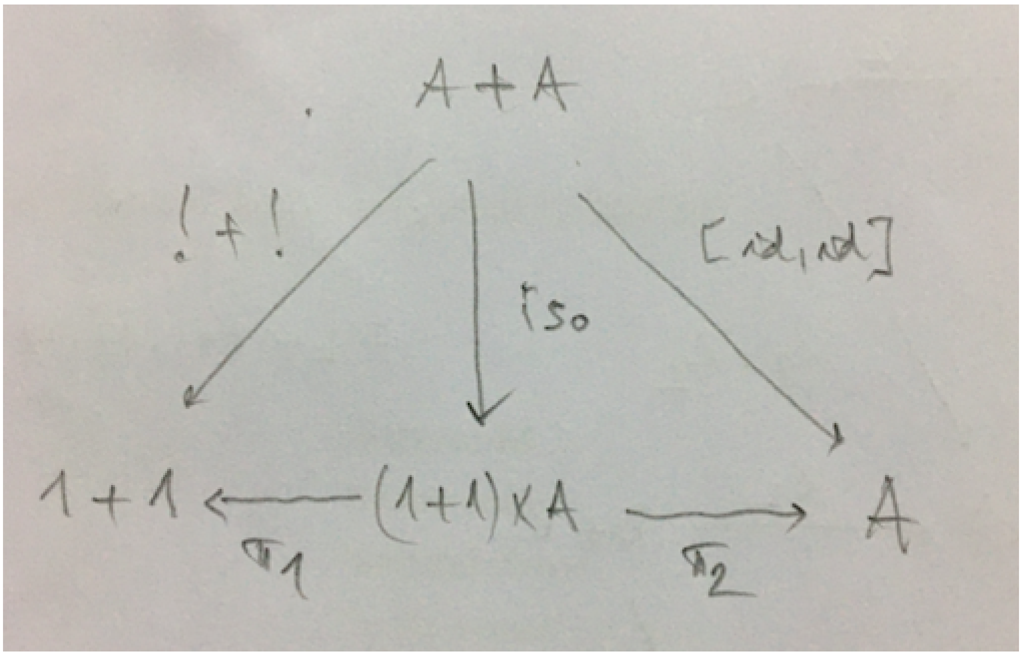
   $$(id \times f) \cdot \mathsf{iso} \quad = \quad \mathsf{iso} \cdot (f + f) \qquad \text{(F1)}$$

   (c) Confirme, por cálculo analítico, essa propriedade.

   (d) Derive uma definição em Haskell *pointwise* de iso.

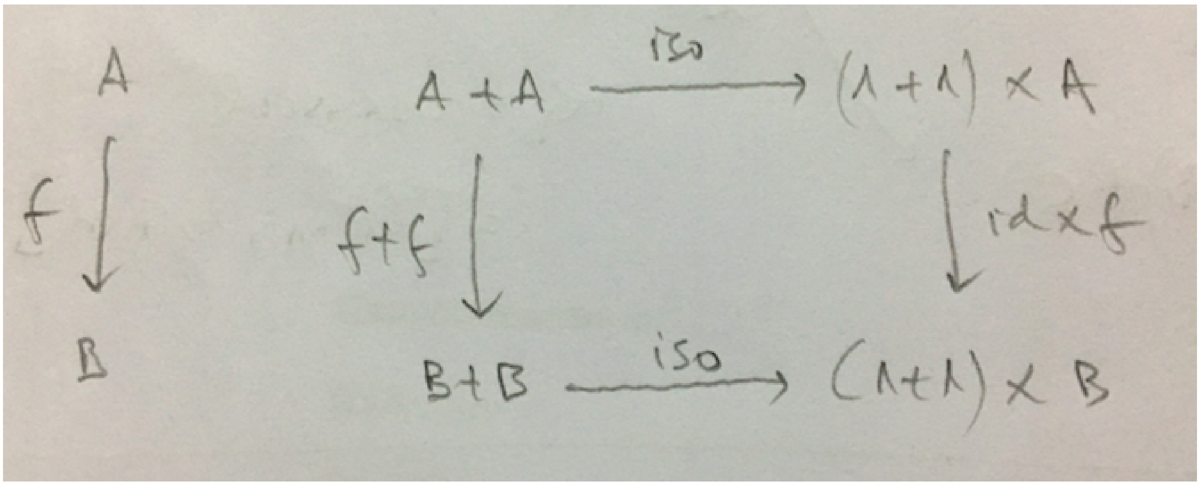
Resolução (a)

$A + A \cong (1 + 1) \times A \cong 2 \times A$



Resolução (b)

$(id \times f)\ .\ iso = iso\ .\ (f + f)$



Resolução (c)

$(id \times f)\ .\ iso = iso\ .\ (f + f)$

Sendo $iso\ =\ <!+!, [id, id]>$ então, pela Lei da Troca, temos:

$(1)\ iso =\ <[i_1.\,!, i_2.\,!], [id, id]>$

$(2)\ iso =\ [<i_1.\,!, id>, <i_2.\,!, id>]$

Vamos então mostrar que

$(id \times f)\ .\ iso = iso\ .\ (f + f)$

{ def-$\times$, **lei (10)** ; def-$+$, **lei (21)** }

$<id.\,\pi_1, f.\,\pi_2>\ .\,iso = iso\ .\,[i_1.\,f, i_2.\,f]$

{ fusão-$\times$, **lei (9)** ; fusão-$+$, **lei (20)** }

$<id.\,\pi_1.\,iso, f.\,\pi_2.\,iso> = [iso.\,i_1.\,f, iso.\,i_2.\,f]$

{ cancelamento-$\times$, **lei(7)** ; $\pi_1.\,iso = [i_1.\,!, i_2.\,!]$; $\pi_2.\,iso = [id, id]$ }

$<id.\,[i_1.\,!, i_2.\,!], f.\,[id, id]> = [iso.\,i_1.\,f, iso.\,i_2.\,f]$

{ cancelamento-$+$, **lei(18)** ; $iso.\,i_1 = <i_1.\,!, id>$; $iso.\,i_2 = <i_2.\,!, id>$ }

$<id.\,[i_1.\,!, i_2.\,!], f.\,[id, id]> = [<i_1.\,!, id>.\,f, <i_2.\,!, id>.\,f]$

{ natural-id, **lei (1)** ; fusão-$\times$, **lei (9)** ; fusão-$+$, **lei (20)** }

$<[i_1.\,!, i_2.\,!], [f, f]> = [<i_1.\,!.\,f, f>, <i_2.\,!.\,f, f>]$

{ def. $! = \underline{()}$; natural-const, **lei (3)** }

$<[i_1.\,!, i_2.\,!], [f, f]> = [<i_1.\,!, f>, <i_2.\,!, f>]$

{ lei da troca, **lei (28)** }

$[<i_1.\,!, f>, <i_2.\,!, f>] = [<i_1.\,!, f>, <i_2.\,!, f>]$

Resolução (d)

$iso = \ \ <!+!, [id, id]>$

{ def-$+$, **lei (21)** }

$\equiv iso = \ \ <[i_1 . !, i_2 . !], [id, id]>$

{ lei da troca, **lei (28)** }

$\equiv iso = \ [<i_1 . !, id>, <i_2 . !, id>]$

{ universal-$+$, $k = iso$ }

$iso \;.\; i_1 = <i_1 . !, id>$

$iso \;.\; i_2 = <i_2 . !, id>$

{ igualdade extensional, **lei (71)** }

$(iso \;.\; i_1)\ a = <i_1 . !, id>\ \ a$

$(iso \;.\; i_2)\ b = <i_2 . !, id>\ \ b$

{ def-comp, **lei (72)**, def-split, **lei (76)** }

$iso\ (i_1\ a) = ((i_1 . !)\ a, id\ a)$

$iso\ (i_2\ b) = ((i_2 . !)\ b, id\ b)$

{ natural-id, **lei (1)**; def-comp, **lei (72)** }

$iso\ (i_1\ a) = (i_1(!\ a), a)$

$iso\ (i_2\ b) = (i_2(!\ b), b)$

{ def. $! = \underline{()}$ }

$iso\ (i_1\ a) = (i_1(), a)$

$iso\ (i_2\ b) = (i_2(), b)$

Haskell

In [1]:

```haskell
iso (Left a) = (Left (),a)
iso (Right b) = (Right (),b)

-- type checking

:t iso
```

```
iso :: forall b. Either b b -> (Either () (), b)
```