Instance of $\rightarrow^+$

$$
\begin{array}{l}
\textbf{newtype}\ a \rightarrow^+ b = \mathit{AddFun}\ (a \rightarrow b) \\
\textbf{instance}\ \mathit{Category}\ (\rightarrow^+)\ \textbf{where} \\
\quad \textbf{type}\ \mathit{Obj}\ (\rightarrow^+) = \mathit{Additive} \\
\quad \mathit{id} = \mathit{AddFun}\ \mathit{id} \\
\quad \mathit{AddFun}\ g \circ \mathit{AddFun}\ f = \mathit{AddFun}\ (g \circ f) \\
\textbf{instance}\ \mathit{Monoidal}\ (\rightarrow^+)\ \textbf{where} \\
\quad \mathit{AddFun}\ f \times \mathit{AddFun}\ g = \mathit{AddFun}\ (f \times g) \\
\textbf{instance}\ \mathit{Cartesian}\ (\rightarrow^+)\ \textbf{where} \\
\quad \mathit{exl} = \mathit{AddFun}\ \mathit{exl} \\
\quad \mathit{exr} = \mathit{AddFun}\ \mathit{exr} \\
\quad \mathit{dup} = \mathit{AddFun}\ \mathit{dup}
\end{array}
$$

Fork and Join

- $\triangledown :: Cartesian\ k \Rightarrow (a\ '\mathrm{k}'\ c) \rightarrow (a\ '\mathrm{k}'\ d) \rightarrow (a\ '\mathrm{k}'\ (c \times d))$
- $\triangle :: Cartesian\ k \Rightarrow (c\ '\mathrm{k}'\ a) \rightarrow (d\ '\mathrm{k}'\ a) \rightarrow ((c + d)\ '\mathrm{k}'\ a)$

...Machine Learning...

Artur Ezequiel Nelson

Universidade do Minho

26 de Abril

Indice

1. Nelson
2. Ezequiel
3. Fork e Join
4. Operacoes Numericas
5. Generalizing Automatic Differentiation
6. Exemplos
7. Generalizar

titulo

titulo

Fork e Join

- (Δ) :: Cartesian k $\Rightarrow$ (a 'k' c) $\rightarrow$ (a 'k' d) $\rightarrow$ (a 'k' (c $\times$ d))
- (∇) :: Cartesian k $\Rightarrow$ (c 'k' a) $\rightarrow$ (d 'k' a) $\rightarrow$ ((c $\times$ d) 'k' a)

instancia de $\rightarrow^+$

newtype a $\rightarrow^+$ b = AddFun (a $\rightarrow$ b)

instance Category ($\rightarrow^+$) **where**
    type Obj ($\rightarrow^+$) = Additive
    id = AddFun id
    AddFun g $\circ$ AddFun f = AddFun (g $\circ$ f)

instance Monoidal ($\rightarrow^+$) **where**
    AddFun f $\times$ AddFun g = AddFun (f $\times$ g)

instance Cartesian ($\rightarrow^+$) **where**
    exl = AddFun exl
    exr = AddFun exr
    dup = AddFun dup

instancia de $\rightarrow^{+}$

instance Cocartesian ($\rightarrow^{+}$) **where**
inl = AddFun inlF
inr = AddFun inrF
jam = AddFun jamF

inlF :: Additive b $\Rightarrow$ a $\rightarrow$ a $\times$ b
inrF :: Additive a $\Rightarrow$ b $\rightarrow$ a $\times$ b
jamF :: Additive a $\Rightarrow$ a $\times$ a $\rightarrow$ a

inlF = λa $\rightarrow$ (a, 0)
inrF = λb $\rightarrow$ (0, b)
jamF = λ(a, b) $\rightarrow$ a + b

definição de NumCat

class NumCat k a **where**

negateC :: a ‘k‘ a

addC :: (a $\times$ a) ‘k‘ a

mulC :: (a $\times$ a) ‘k‘ a

...

instance Num a $\Rightarrow$ NumCat ($\rightarrow$) a **where**

negateC = negate

addC = uncurry (+)

mulC = uncurry ($\cdot$)

...

$$D\ (\text{negate}\ u) = \text{negate}\ (D\ u)$$
$$D\ (u + v) = D\ u + D\ v$$
$$D\ (u \cdot v) = u \cdot D\ v + v \cdot D\ u$$

- Impreciso na natureza de u e v.
- Algo mais preciso seria defenir a diferenciação das operações em si.

class Scalable k a **where**
    scale :: a $\rightarrow$ (a ‘k‘ a)

instance Num a $\Rightarrow$ Scalable ($\rightarrow^{+}$) a **where**
    scale a = AddFun (λda $\rightarrow$ a $\cdot$ da)

instance NumCat D **where**
    negateC = linearD negateC
    addC = linearD addC
    mulC = D (λ(a, b) $\rightarrow$ (a $\cdot$ b, scale b ∇ scale a))

Generalizing Automatic Differentiation

newtype D_k a b = D (a $\rightarrow$ b $\times$ (a ‘k‘ b))

linearD :: (a $\rightarrow$ b) $\rightarrow$ (a ‘k‘ b) $\rightarrow$ D_k a b
linearD f f’= D (λa $\rightarrow$ (f a, f’))

instance Category k $\Rightarrow$ Category D_k **where**
type Obj D_k = Additive $\wedge$ Obj k ...

instance Monoidal k $\Rightarrow$ Monoidal D_k **where** ...

instance Cartesian k $\Rightarrow$ Cartesian D_k **where** ...

instance Cocartesian k $\Rightarrow$ Cocartesian D_k **where**
inl = linearD inlF inl
inr = linearD inrF inr
jam = linearD jamF jam

instance Scalable k s $\Rightarrow$ NumCat D_k s **where**

negateC = linearD negateC negateC

addC = linearD addC addC

mulC = D (λ(a, b) $\rightarrow$ (a $\cdot$ b, scale b ∇ scale a))

Exemplos

Generalizar