...Machine Learning...

Artur Ezequiel Nelson

Universidade do Minho

26 de Abril

Indice

1. Nelson
2. Categorias
3. Fork e Join
4. Operacoes Numericas
5. Exemplos
6. Generalizar

titulo

Uma curta introdução

- Queremos calcular $\mathcal{D}^+$.
- Problema: $\mathcal{D}$ não é computável.
- Solução: observar corolários apresentados e implementar recorrendo a categorias.

Uma curta introdução

- Queremos calcular $\mathcal{D}^+$.
- Problema: $\mathcal{D}$ não é computável.
- Solução: observar corolários apresentados e implementar recorrendo a categorias.

Uma curta introdução

- Queremos calcular $\mathcal{D}^{+}$.
- Problema: $\mathcal{D}$ não é computável.
- Solução: observar corolários apresentados e implementar recorrendo a categorias.

Uma curta introdução

Corolário 1.1

NOTA: adicionar definição do corolário 1.1 aqui

Corolário 2.1

NOTA: adicionar definição do corolário 2.1 aqui

Corolário 3.1

NOTA: adicionar definição do corolário 3.1 aqui

Categorias clássicas

Uma categoria é um conjunto de objetos(conjuntos e tipos) e de morfismos(operações entre objetos), tendo definidas 2 operações básicas, identidade e composição de morfismos, e 2 leis:

- (C.1) —- $id \circ f = id \circ f = f$
- (C.2) —- $f \circ (g \circ h) = (f \circ g) \circ h$

Para os efeitos deste papel, objetos são tipos de dados e morfismos são funções

class *Category k* where
 id :: (a'k'a)
 ($\circ$) :: (b'k'c) $\rightarrow$ (a'k'b) $\rightarrow$ (a'k'c)

instance *Category* $(\rightarrow)$ where
 id = λa $\rightarrow$ a
 $g \circ f = \lambda$a $\rightarrow$ g (f a)

Categorias clássicas

Uma categoria é um conjunto de objetos(conjuntos e tipos) e de morfismos(operações entre objetos), tendo definidas 2 operações básicas, identidade e composição de morfismos, e 2 leis:

- (C.1) —- $id \circ f = id \circ f = f$
- (C.2) —- $f \circ (g \circ h) = (f \circ g) \circ h$

Para os efeitos deste papel, objetos são tipos de dados e morfismos são funções

class *Category k* where
id :: (a'k'a)
($\circ$) :: (b'k'c) $\rightarrow$ (a'k'b) $\rightarrow$ (a'k'c)

instance *Category ($\rightarrow$)* where
id = λa $\rightarrow$ a
$g \circ f = \lambda$a $\rightarrow$ g (f a)

Functores clássicos

Um functor F entre categorias $\mathcal{U}$ e $\mathcal{V}$ é tal que:

- para qualquer objeto t $\in \mathcal{U}$ temos que F t $\in \mathcal{V}$
- para qualquer morfismo m :: a $\rightarrow$ b $\in \mathcal{U}$ temos que F m :: F a $\rightarrow$ F b $\in \mathcal{V}$
- F id $(\in \mathcal{U})$ = id $(\in \mathcal{V})$
- F $(f \circ g)$ = F f $\circ$ F g

Nota

Devido à definição de categoria deste papel(objetos são tipos de dados) os functores mapeiam tipos neles próprios.

Objetivo

Começamos por definir um novo tipo de dados:
newtype $\mathcal{D}$ a b = $\mathcal{D}(a \to b \times (a \multimap b))$
Depois adaptamos $\mathcal{D}^+$ para usar este tipo de dados:

Definição adaptada

$\hat{\mathcal{D}}$ :: (a $\to$ b) $\to$ $\mathcal{D}$ a b
$\hat{\mathcal{D}}$ f = $\mathcal{D}(\mathcal{D}^+$ f)

O nosso objetivo é a dedução de uma instância de categoria para $\mathcal{D}$ onde $\hat{\mathcal{D}}$ seja functor.

Dedução da instância

Recordando os corolários 3.1 e 1.1 deduzimos que

- (DP.1) —- $\mathcal{D}^{+} id$ = λa $\rightarrow$ (id a,id)
- (DP.2) —- $\mathcal{D}^{+}(g \circ f)$ = $\lambda a \rightarrow let\{(b, f')$ = $\mathcal{D}^{+}$ f a; $(c, g') = \mathcal{D}^{+}$ g b $\}$ in $(c, g' \circ f')$

$\hat{\mathcal{D}}$ ser functor é equivalente a dizer que, para todas as funções f e g de tipos apropriados:

- id = $\hat{\mathcal{D}}$ id = $\mathcal{D}(\mathcal{D}^{+} id)$
- $\hat{\mathcal{D}}$ g $\circ$ $\hat{\mathcal{D}}$ f = $\hat{\mathcal{D}}$ (g $\circ$ f) = $\mathcal{D}(\mathcal{D}^{+}(g \circ f))$

Dedução da instância

Com base em (DP.1) e (DP.2) podemos reescrever como sendo:

- id = $\mathcal{D}(\lambda$a $\rightarrow$ (id a,id))
- $\hat{\mathcal{D}}$ g $\circ$ $\hat{\mathcal{D}}$ f = $\mathcal{D}$ ($\lambda a \rightarrow let\{(b, f')$ = $\mathcal{D}^{+}$ f a; $(c, g') = \mathcal{D}^{+}$ g b } in $(c, g' \circ f')$)

Resolver a primeira equação é trivial(definir id da instância como sendo $\mathcal{D}(\lambda$a $\rightarrow$ (id a,id))).
A segunda equação será resolvida resolvendo uma condição mais geral: $\mathcal{D}g \circ \mathcal{D}f$ = $\mathcal{D}(\lambda a \rightarrow let\{(b, f')$ = f a; (c, g') = g b } in $(c, g' \circ f')$), cuja solução é igualmente trivial.

Dedução da instância

Definição de $\hat{\mathcal{D}}$ para funções lineares

linearD :: (a $\rightarrow$ b) $\rightarrow$ $\mathcal{D}$ a b
linearD f = $\mathcal{D}(\lambda$a $\rightarrow$ (f a,f))

Instância da categoria que deduzimos

instance *Category* $\mathcal{D}$ where
  id = linearD id
  $\mathcal{D}g \circ \mathcal{D}f$ = $\mathcal{D}(\lambda a \rightarrow let\{(b, f') =$ f a; $(c, g') =$ g b } in $(c, g' \circ f'))$

Prova da instância

Antes de continuarmos devemos verificar se esta instância obedece às leis (C.1) e (C.2).
Se considerarmos apenas morfismos $\hat{f}$ :: $\mathcal{D}$ a b tal que $\hat{f}$ = $\mathcal{D}^+$ f para f :: a $\rightarrow$ b(o que podemos garantir se transformarmos $\mathcal{D}$ a b em tipo abstrato) podemos garantir que $\mathcal{D}^+$ é functor.

Prova de (C.1)

id $\circ\hat{\mathcal{D}}$
= $\hat{\mathcal{D}}id \circ \hat{\mathcal{D}}$ f -lei functor de id (especificação de $\hat{\mathcal{D}}$)
= $\hat{\mathcal{D}}$ (id $\circ$ f) - lei functor para ($\circ$)
= $\hat{\mathcal{D}}$ f - lei de categoria

Prova da instância

Prova de (C.2)

$\hat{\mathcal{D}}$ h $\circ$ ($\hat{\mathcal{D}}$ g $\circ$ $\hat{\mathcal{D}}$ f)
= $\hat{\mathcal{D}}$ h $\circ$ $\hat{\mathcal{D}}$ (g $\circ$ f) - lei functor para ($\circ$)
= $\hat{\mathcal{D}}$ (h $\circ$ (g $\circ$ f)) - lei functor para ($\circ$)
= $\hat{\mathcal{D}}$ ((h $\circ$ g) $\circ$ f) - lei de categoria
= $\hat{\mathcal{D}}$ (h $\circ$ g) $\circ$ $\hat{\mathcal{D}}$ f - lei functor para ($\circ$)
= ($\hat{\mathcal{D}}$ h $\circ$ $\hat{\mathcal{D}}$ g) $\circ$ $\hat{\mathcal{D}}$ f - lei functor para ($\circ$)

Nota

Estas provas não requerem nada de $\mathcal{D}$ e $\hat{\mathcal{D}}$ para além das leis do functor, logo nas próximas instâncias deduzidas de um functor não precisamos de voltar a realizar estas provas.

Categorias e functores monoidais

A versão generalizada da composição paralela será definida através de uma categoria monoidal:

class *Category k* $\Rightarrow$ *Monoidal k* where

$(\times)$::(a'k'c)$\rightarrow$(b'k'd)$\rightarrow$((a$\times$b)'k'(c$\times$d))

instance *Monoidal* $(\rightarrow)$ where

$f \times g = \lambda$(a,b)$\rightarrow$(f a,g b)

Definição de functor monoidal

Um functor *F* monoidal entre categorias $\mathcal{U}$ e $\mathcal{V}$ é tal que:

- *F* é functor clássico
- *F* (f $\times$ g) = *F* f $\times$ *F* g

Dedução da instância

A partir do corolário 2.1 deduzimos que:
$\mathcal{D}^+$ (f $\times$ g) = λ(a,b) $\rightarrow$ let{(c,f')= $\mathcal{D}^+$ f a; (d,g') = $\mathcal{D}^+$ g b } in ((c,d),f'$\times$g')
Se definirmos o functor F a partir de $\hat{\mathcal{D}}$ chegamos à seguinte condição:
$\mathcal{D}(\mathcal{D}^+$ f) $\times$ $\mathcal{D}(\mathcal{D}^+$ g) = $\mathcal{D}(\mathcal{D}^+$ (f $\times$ g))
Substituindo e fortalecendo-a obtemos:
$\mathcal{D}$ f $\times$ $\mathcal{D}$ g = $\mathcal{D}(\lambda$(a,b) $\rightarrow$ let{(c,f') = f a; (d,g') = g b } in ((c,d),f'$\times$g'))
e esta condição é suficiente para obtermos a nossa instância.

Dedução da instância

Instância da categoria que deduzimos

instance *Monoidal* $\mathcal{D}$ where
$\mathcal{D}$ f $\times$ $\mathcal{D}$ g = $\mathcal{D}(\lambda$(a,b) $\rightarrow$ let{(c,f') = f a; (d,g') = g b } in ((c,d),f'$\times$g'))

Categorias e funtores cartesianas

class *Monoidal k* $\Rightarrow$ *Cartesean k* where
 exl :: (a$\times$b)'k'a
 exr :: (a$\times$b)'k'b
 dup :: a'k'(a$\times$a)

instance *Cartesean* $(\rightarrow)$ where
 exl = λ(a,b) $\rightarrow$ a
 exr = λ(a,b) $\rightarrow$ b
 dup = λa $\rightarrow$ (a,a)

Um functor *F* cartesiano entre categorias $\mathcal{U}$ e $\mathcal{V}$ é tal que:

- *F* é functor monoidal
- *F* exl = exl
- *F* exp = exp
- *F* dup = dup

Dedução da instância

Pelo corolário 3.1 e pelo facto que exl,exr e dup são linerares deduzimos que:

$\mathcal{D}^+$ exl λp $\rightarrow$ (exp p, exl)

$\mathcal{D}^+$ exr λp $\rightarrow$ (exr p, exr)

$\mathcal{D}^+$ dup λa $\rightarrow$ (dup a, dup)

Após esta dedução podemos continuar a determinar a instância:

exl = $\mathcal{D}(\mathcal{D}^+$ exl)

exr = $\mathcal{D}(\mathcal{D}^+$ exr)

dup = $\mathcal{D}(\mathcal{D}^+$ dup)

Dedução da instância

Substituindo e usando a definição de linearD obtemos:
exl = linearD exl
exr = linearD exr
dup = linearD dup
E podemos converter a dedução acima diretamente em instância:

Instância da categoria que deduzimos

instance *Cartesian* $\mathcal{D}$ where
  exl = linearD exl
  exr = linearD exr
  dup = linearD dup

Categorias cocartesianas

São o dual das categorias cartesianas.

Nota

Neste papel os coprodutos correspondem aos produtos das categorias, i.e., categorias de biprodutos.

class *Category k* $\Rightarrow$ *Cocartesian k* where:
inl :: a'k'(a$\times$b)
inlr:: b'k'(a$\times$b)
jam :: (a$\times$a)'k'a

Functores cocartesianos

Definição de functor cocartesiano

Um functor F cartesiano entre categorias $\mathcal{U}$ e $\mathcal{V}$ é tal que:

- F é functor
- F inl = inl
- F inr = inr
- F jam = jam

Fork e Join

- $(\triangle)$ :: Cartesian k $\Rightarrow$ (a 'k' c) $\rightarrow$ (a 'k' d) $\rightarrow$ (a 'k' (c $\times$ d))
- $(\triangledown)$ :: Cartesian k $\Rightarrow$ (c 'k' a) $\rightarrow$ (d 'k' a) $\rightarrow$ ((c $\times$ d) 'k' a)
- **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

Fork e Join

- $(\triangle)$ :: Cartesian k $\Rightarrow$ (a 'k' c) $\rightarrow$ (a 'k' d) $\rightarrow$ (a 'k' (c $\times$ d))
- $(\triangledown)$ :: Cartesian k $\Rightarrow$ (c 'k' a) $\rightarrow$ (d 'k' a) $\rightarrow$ ((c $\times$ d) 'k' a)
- **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

Operações Numéricas

- ola

Exemplos

Generalizar