$$(f\ x\ g)\ (a, b) = (f\ a, g\ b)$$

$$(f = \perp)$$

Instance of $\rightarrow^{+}$

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**

  $exl = AddFun\ exl$

  $exr = AddFun\ exr$

  $dup = AddFun\ dup$

Fork and Join

- $\nabla :: Cartesian\ k \Rightarrow (a \ '\mathtt{k}'\ c) \rightarrow (a \ '\mathtt{k}'\ d) \rightarrow (a \ '\mathtt{k}'\ (c \times d))$
- $\Delta :: Cartesian\ k \Rightarrow (c \ '\mathtt{k}'\ a) \rightarrow (d \ '\mathtt{k}'\ a) \rightarrow ((c + d) \ '\mathtt{k}'\ a)$

...Machine Learning...

Artur Ezequiel Nelson

Universidade do Minho

26 de Abril

Indice

A short introdution

- We want to calculate $\mathcal{D}$.
- However, $\mathcal{D}$ is not computable.
- Solution: reimplement corollaries using category teory

A short introduction

- We want to calculate $\mathcal{D}$.
- However, $\mathcal{D}$ is not computable.
- Solution: reimplement corollaries using category teory

A short introdution

- We want to calculate $\mathcal{D}$.
- However, $\mathcal{D}$ is not computable.
- Solution: reimplement corollaries using category teory

A short introdution

Corollary 1.1

NOTA: adicionar definição do corolário 1.1 aqui

Corollary 2.1

NOTA: adicionar definição do corolário 2.1 aqui

Corollary 3.1

NOTA: adicionar definição do corolário 3.1 aqui

Categories

A category is a collection of objects(sets and types) e morphisms(operation between objects), with 2 basic operations(identity and composition) of morfisms, and 2 laws:

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

N

ote: for this paper, objects are data types and morfisms are functions

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

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

Functors

A functor F between 2 categories $\mathcal{U}$ *and* $\mathcal{V}$ is such that:

- given any object $t\lambda$**in** $\mathcal{U}$ there exists a object $F\ t\lambda$**in** $\mathcal{V}$
- given any morphism $m :: a \to b\lambda$**in** $\mathcal{U}$ there exists a morphism $F\ m :: F\ a \to F\ b\lambda$**in** $\mathcal{V}$
- $F\ id\ (\lambda$**in** $\mathcal{U}) = id\ (\lambda$**in** $\mathcal{V})$
- $F\ (f\ \$\backslash\ circ\ \$\ g) = F\ f\ \$\backslash\ circ\ \$\ F\ g$

Note

Given this papers category properties(objects are data types) we have that functors map types to themselfs

Objective

Let's start by defining a new data type:
newtype $\mathcal{D}\ a\ b = \mathcal{D}\ (a \to b\ x\ (a\,\$\backslash\ multimap\,\$\ b))$
and adapting $\mathcal{D}^+$ to use it:

Adapted definition

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

Our objective is to deduce an instance of Category for $\mathcal{D}$ **where** $\hat{\mathcal{D}}$ is a functor.

Instance deduction

Using corollaries 3.1 and 1.1 we deduce that

- (DP.1) -- bigDplus id = λ a -> (id a,id)
- (DP.2) —- $\mathcal{D}^+\ (g\,\$\backslash\ circ\,\$\ f) = \$\backslash lambda\,\$\ a \to$ **let** $\{(b, f') = \mathcal{D}^+\ f\ a; (c, g') = \mathcal{D}^+\ g\ b\}$ **in** $(c, g'\ \$\backslash\ circ\,\$\ f')$

saying that $\hat{\mathcal{D}}$ *a functor is equivalent to, for all f e g functions* **of** *corre*
$id = \hat{\mathcal{D}}\ id = \mathcal{D}\ (\mathcal{D}^+\ id)$
$\hat{\mathcal{D}}\ g\,\$\backslash\ circ\,\$\ \hat{\mathcal{D}}\ f = \hat{\mathcal{D}}\ (g\,\$\backslash\ circ\,\$\ f) = \mathcal{D}\ (\hat{\mathcal{D}}\ (g\,\$\backslash\ circ\,\$\ f))$

Instance deduction

Based on (DP.1) e (DP.2) we'll rewrite the above into the following defenition:
$id = \mathcal{D}\ (\$\backslash lambda\,\$\ a \rightarrow (id\ \ a, id))$
$bidDhat\ g\,\$\backslash\ circ\,\$\,\hat{\mathcal{D}}\ f = \mathcal{D}\ (\$\backslash lambda\,\$\ a \rightarrow \textbf{let}\ \{(b, f') = \mathcal{D}^{+}\ f\ a; (c, g') = \mathcal{D}^{+}\ g\ b\}\ \textbf{in}\ (c, g'\ \$\backslash\ circ\,\$\ f'))$
The first equasion has a trivial solution(define id of instance as $\mathcal{D}\ (\$\backslash lambda\,\$\ a \rightarrow (id\ \ a, id)))$
To solve the secound we'll first solve a more general one:
$\mathcal{D}\ g\,\$\backslash\ circ\,\$\,\mathcal{D}\ f = \mathcal{D}\ (\$\backslash lambda\,\$\ a \rightarrow \textbf{let}\ \{(b, f') = f\ \ a; (c, g') = g\ b\lambda\}\ \textbf{in}\ (c, g'\ \$\backslash\ circ\,\$\ f'))$, and this has an equivalently trivial solution in our instance.

Instance deduction

$\hat{\mathcal{D}}$ definition for linear functions

$$\begin{aligned}&\mathit{linearD} :: (a \rightarrow b) \rightarrow \mathcal{D}\ a\ b\\&\mathit{linearD}\ f = \mathcal{D}\ (\lambda a \rightarrow (f\ a, f))\end{aligned}$$

s

Categorical instance we've deduced

$$\begin{aligned}&\textbf{instance}\ \mathit{Category}\ \mathcal{D}\ \textbf{where}\\&\quad \mathit{id} = \mathit{linearD}\ \mathit{id}\\&\quad \mathcal{D}\ g \circ \mathcal{D}\ f = \mathcal{D}\ (\lambda a \rightarrow \textbf{let}\ \{(b, f') = f\ a; (c, g') = g\ b\}\ \textbf{in}\ (c, g' \circ\end{aligned}$$

Instance proof

In order to prove that the instance is correct we must observe if it follows laws (C.1) and (C.2).
First we must make a concession: that we only use morfisms arising from $\mathcal{D}^+$(we can force this by transforming $\mathcal{D}$ into an abstract type). If we do, then $\mathcal{D}^+$ is a functor.

(C.1) proof

$id\ \$\backslash\ circ\ \$\ \hat{\mathcal{D}}$
$= \hat{\mathcal{D}}\ id\ \$\backslash\ circ\ \$\ \hat{\mathcal{D}}\ f - functor\ law\ for\ id\ (specification\ \textbf{of}\ \hat{\mathcal{D}})$
$= \hat{\mathcal{D}}\ (id\ \$\backslash\ circ\ \$\ f) - functor\ law\ for\ (\$\backslash circ\$)$
$= \hat{\mathcal{D}}\ f - cathegorical\ law$

Instance proof

(C.2) proof

$\hat{\mathcal{D}}\ h\,\$\backslash\ circ\,\$\ (\hat{\mathcal{D}}\ g\,\$\backslash\ circ\,\$\ \hat{\mathcal{D}}\ f)$
$= \hat{\mathcal{D}}\ h\,\$\backslash\ circ\,\$\ \hat{\mathcal{D}}\ (g\,\$\backslash\ circ\,\$\ f)$ – *functor law for* $(\$\backslash circ\$)$
$= \hat{\mathcal{D}}\ (h\,\$\backslash\ circ\,\$\ (g\,\$\backslash\ circ\,\$\ f))$ – *functor law for* $(\$\backslash circ\$)$
$= \hat{\mathcal{D}}\ ((h\,\$\backslash\ circ\,\$\ g)\,\$\backslash\ circ\,\$\ f)$ – *categorical law*
$= \hat{\mathcal{D}}\ (h\,\$\backslash\ circ\,\$\ g)\,\$\backslash\ circ\,\$\ \hat{\mathcal{D}}\ f$ – *functor law for* $(\$\backslash circ\$)$
$= (\hat{\mathcal{D}}\ h\,\$\backslash\ circ\,\$\ \hat{\mathcal{D}}\ g)\,\$\backslash\ circ\,\$\ \hat{\mathcal{D}}\ f$ – *functor law for* $(\$\backslash circ\$)$

Note

This proofs don't require anything from $\mathcal{D}$ *and* $\hat{\mathcal{D}}$ aside from functor laws. As such, all other instances of categories created from a functor won't require further proofs.

Monoidal categories and functors

Generalized parallel composition will be defined using a monoidal category:

class *Category k* $\Rightarrow$ *Monoidal k* **where**
$(\times) :: (a \text{ `k` } c) \rightarrow (b \text{ `k` } d) \rightarrow ((a \times b) \text{ `k` } (c \times d))$

instance *Monoidal* $(\rightarrow)$ **wh**
$f \times g = \lambda(a, b) \rightarrow (f\ a, g$

Monoidal Functor definition

A monoidal functor F between categories $\mathcal{U}$ *and* $\mathcal{V}$ is such that:

- F is a functor
- F (f $\backslash$*times*$ g) = F f $\backslash$*times*$ F g

Instance deduction

From corollary 2.1 we can deduce that:
$\mathcal{D}^{+}\ (f\ \$\backslash\ times\ \$\ g) = \$\backslash lambda\ \$\ (a, b) \rightarrow \textbf{let}\ \{(c, f') = \mathcal{D}^{+}\ f\ a; (d, g') = \mathcal{D}^{+}\ g\ b\}\ \textbf{in}\ ((c, d), f'\ \$\backslash\ times\ \$\ g')$
Defining F from $\hat{\mathcal{D}}$ leaves us with the following definition:
$\mathcal{D}\ (\mathcal{D}^{+}\ f)\ \$\backslash\ times\ \$\ \mathcal{D}\ (\mathcal{D}^{+}\ g) = \mathcal{D}\ (\mathcal{D}^{+}\ (f\ \$\backslash\ times\ \$\ g))$
Using the same method as before, we replace $\mathcal{D}^{+}$ with it's definition and generalize the condition:
$\mathcal{D}\ f\ \$\backslash\ times\ \$\ \mathcal{D}\ g = \mathcal{D}\ (\$\backslash lambda\ \$\ (a, b) \rightarrow \textbf{let}\ \{(c, f') = f\ a; (d, g') = g\ b\}\ \textbf{in}\ ((c, d), f'\ \$\backslash\ times\ \$\ g'))$
and this is enouth for our new instance.

Instance deduction

Categorical instance we've deduced

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

Cartesian categories and functors

$$\textbf{class } \mathit{Monoidal}\ k \Rightarrow \mathit{Cartesean}\ k\ \textbf{where}$$
$$\mathit{exl} :: (a, b)\ \texttt{'k'}\ a$$
$$\mathit{exr} :: (a, b)\ \texttt{'k'}\ b$$
$$\mathit{dup} :: a'k'\ (a, a)$$

$$\textbf{instance } \mathit{Cartesean}\ (\rightarrow)\ \textbf{w}$$
$$\mathit{exl} = \lambda(a, b) \rightarrow a$$
$$\mathit{exr} = \lambda(a, b) \rightarrow b$$
$$\mathit{dup} = \lambda a \rightarrow (a, a)$$

A cartesian functor F between categories $\mathcal{U}$ *and* $\mathcal{V}$ is such that:

- F is a monoidal functor
- F exl = exl
- F exp = exp
- F dup = dup

Instance deduction

From corollary 3.1 and from exl,exr and dup beeing linear function we can deduce that:
$\mathcal{D}^+ \; exl \lambda p \rightarrow (exl\ p, exl)$ $\mathcal{D}^+ \; exr \lambda p \rightarrow (exr\ p, exr)$
$\mathcal{D}^+ \; dup \lambda p \rightarrow (dup\ a, dup)$
With this in mind we'll deduce the instance: exl = $\mathcal{D}$ $(\mathcal{D}^+ \; exl)$
exr = $\mathcal{D}$ $(\mathcal{D}^+ \; exr)$ dup = $\mathcal{D}$ $(\mathcal{D}^+ \; dup)$

Instance deduction

Replacing $\mathcal{D}^+$ with it's definition and remembering linearD we can obtain:
exl = linearD exl exr = linearD exr dup = linearD dup
and we can directly convert this into a new instance:

Categorical instance we've deduced

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

Cocartesian category

This type of categories are the dual of the cartesian categories.

Note

In this paper coproducts are categorical products, i.e., biproducts

Definition

$$
\begin{array}{l}
\textbf{class } \mathit{Category}\ k \Rightarrow \mathit{Cocartesian}\ k \textbf{ where}: \\
\quad \mathit{inl} :: a'k'\ (a, b) \\
\quad \mathit{inr} :: b'k'\ (a, b) \\
\quad \mathit{jam} :: (a, a)\ '\mathtt{k}'\ \ a
\end{array}
$$

Cocartesian functors

Cocartesian functor definition

A cocartesian functor F between categories $\mathcal{U}$ *and* $\mathcal{V}$ is such that:

- F is a functor
- F inl = inl
- F inr = inr
- F jam = jam