All of Basic Categories

Mark Hopkins
@antiselfdual
mjhopkins.github.io

GRUPPENPEST

GRUPPENPEST

Group theory

$$p^+ \; n^0 \; e^-$$

$$e^+$$

$$\mu^-$$

$$\pi^+ \; \pi^0 \; \pi^-$$

$$K^+ \; K^- \; K^0 \; \bar{K}^0$$

$$\Lambda^0$$

$$\Delta^{++} \; \Delta^+ \; \Delta^0 \; \Delta^-$$

$$\Sigma^+ \; \Sigma^0 \; \Sigma^-$$

$$\Xi^+ \; \Xi^0 \; \Xi^-$$

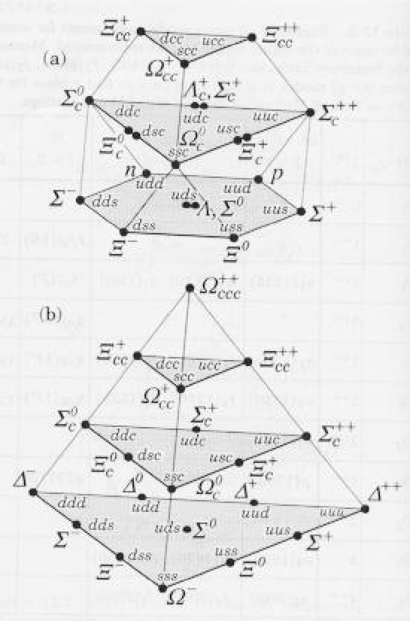
$$\Sigma^{*+} \; \Sigma^{*0} \; \Sigma^{*-}$$

$$\Xi^0 \; \Xi^-$$

$$\Xi^{*0} \; \Xi^{*-}$$

$$\vdots$$

Eightfold way



Is CT the Gruppenpest of FP?

Is CT the Gruppenpest of FP?

- something new to learn
- abstract
- a little scary
- a good model
- helps explain what's going on
- gives us new tools

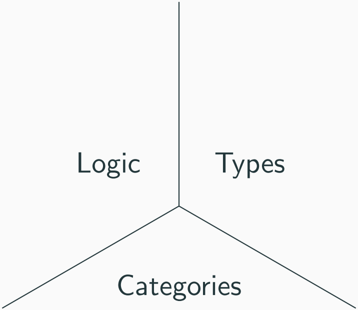
Curry-Howard correspondence

Logic	Programming
proposition	type
proof	program

Logic	Programming
F	Void
T	()
$p \Rightarrow q$	p $\rightarrow$ q
$\neg p$	p $\rightarrow$ Void
$p \wedge q$	(p,q)
$p \vee q$	Either p q

Curry-Howard correspondence

Curry-Howard-Lambek correspondence



"Computational Trinitarianism"

type theory $\rightarrow$ categorical model

internal language $\leftarrow$ category

Preliminaries

CAUTION

Ride moves quickly and makes sharp turns. Please keep arms and legs within the car and keep your seatbelt fastened.

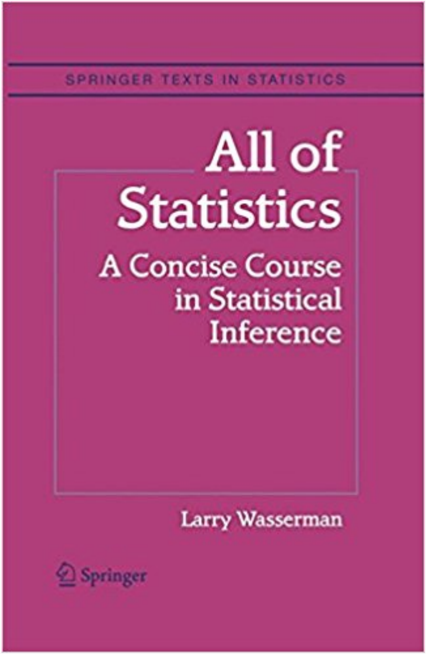
I've *used* category theory. . .

But I'm not a category theorist.

All of?

The impossible is the only thing worth attempting.

Raimondo Panikkar

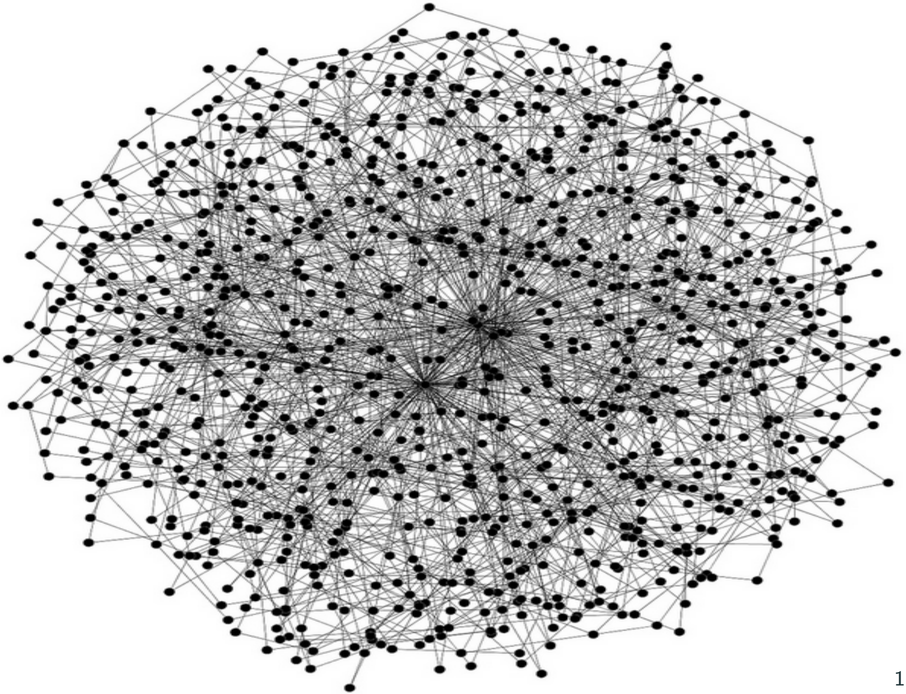

SPRINGER TEXTS IN STATISTICS
All of Statistics
A Concise Course in Statistical Inference
Larry Wasserman
Springer

So we’ll be experts in 30 mins?

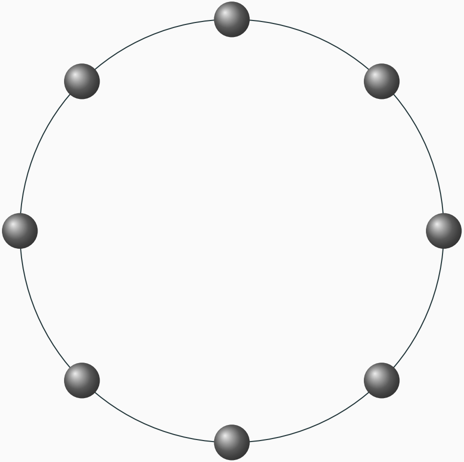
All of
Basic
Categories

ABCs

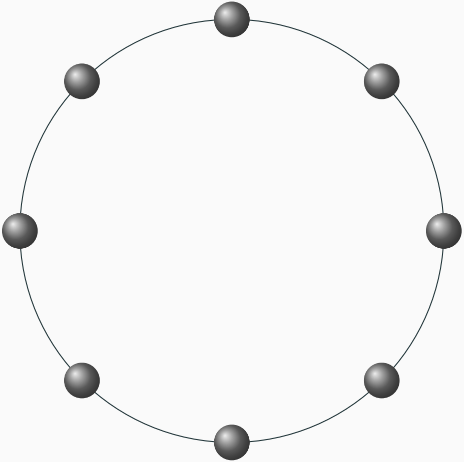

Distributive law
Codensity monad
Profunctor
Lawvere theory
Representable
Contravariant
Yoneda embedding
Kleisli category
Isomorphism
Kan extension
Cauchy completion
Coalgebra
Monoidal category
Grothendieck fibration
Functor
F-algebra
Gray category
T-algebra
Lax monoidal functor
Eilenberg-Moore category
Coend
Internal hom
Topos
Comonoid
Cayley
Day convolution
Free
T-span
Presheaf
cospan
Cofree
Operad
span
Tambara module
Algebra for a monad
combinatorial species
Yoneda lemma
Galois connection
End
Cartesian closed
Adjunction
Monad
Comonad
Density comonad



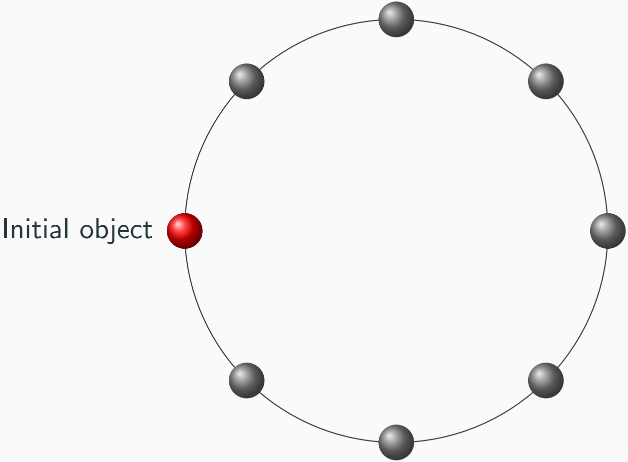
Circle of life



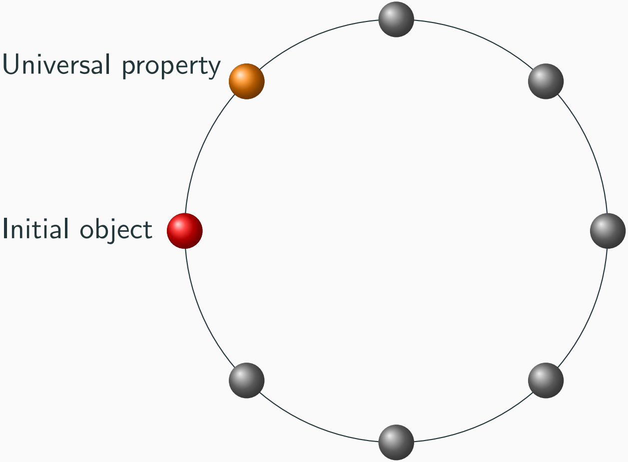
Circle of ~~life~~ universal constructions



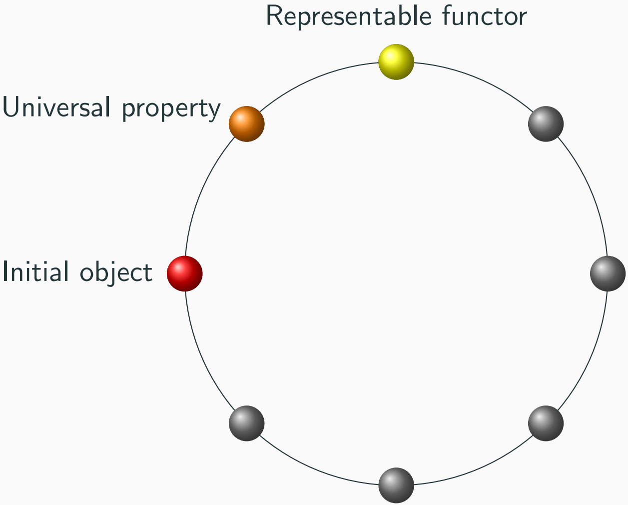
Circle of ~~life~~ universal constructions



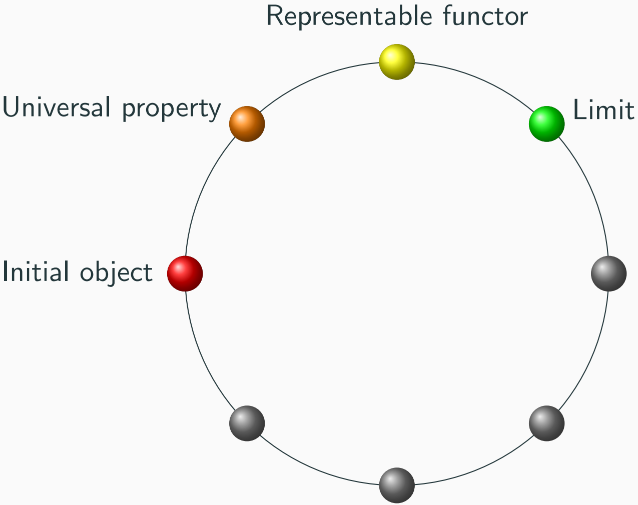
Circle of ~~life~~ universal constructions



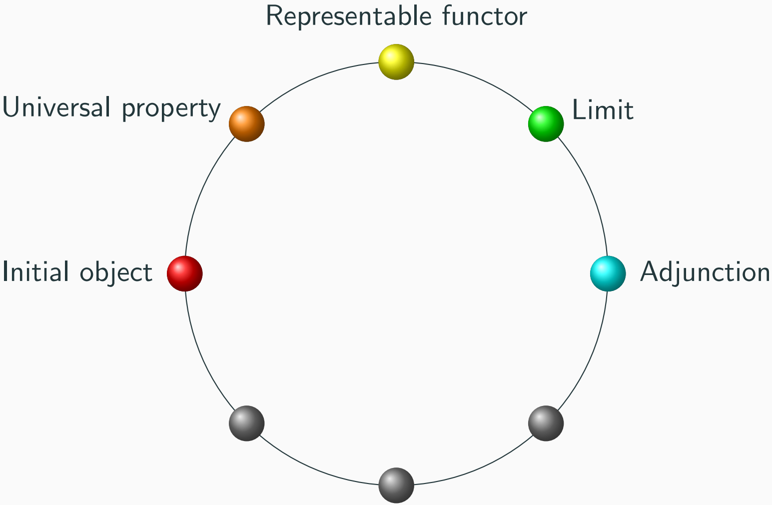
Circle of ~~life~~ universal constructions



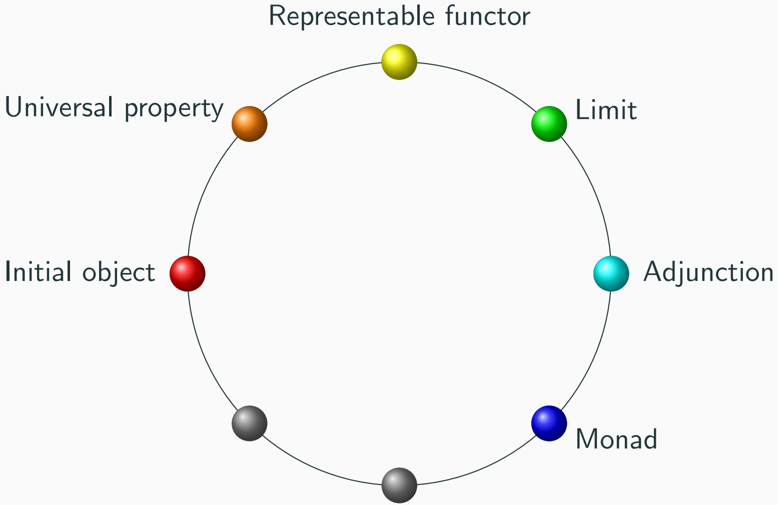
Circle of ~~life~~ universal constructions



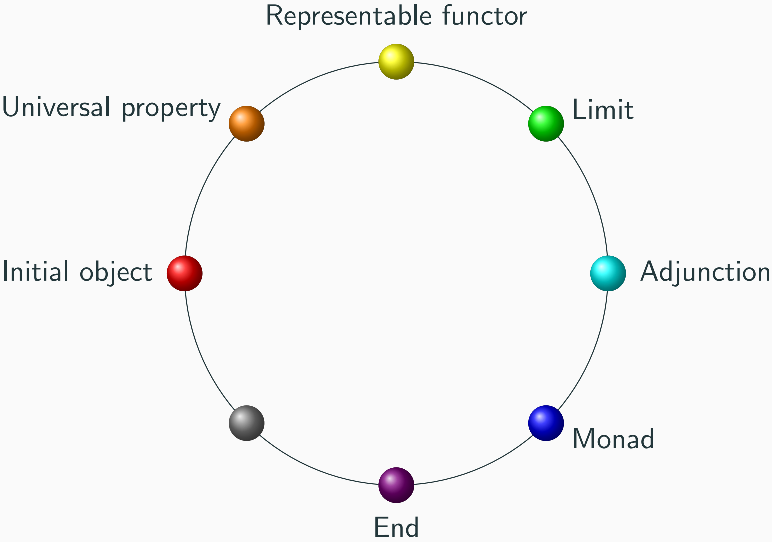
Circle of ~~life~~ universal constructions



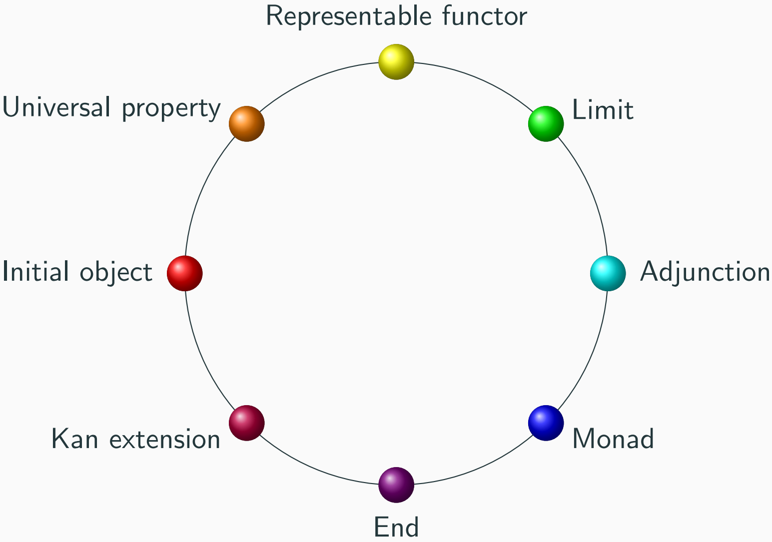
Circle of ~~life~~ universal constructions



Circle of ~~life~~ universal constructions

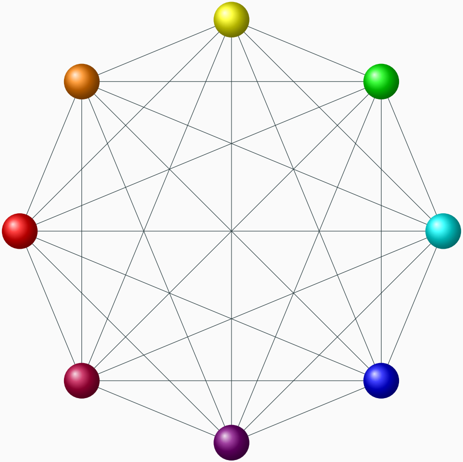


Circle of ~~life~~ universal constructions




Basic category theory
Representable functor
Presheaf
Yoneda Lemma
Limit
Equalizer
Pushout
Product
Adjunctions
Galois connection
Free
Cartesian closed category
Monad
Algebras for a monad
Kleisli category
Eilenberg Moore category
End
End calculus
Kan extension
Density monad
Day convolution
Initial object
F-algebra
Universal property
Universal element
Universal arrow

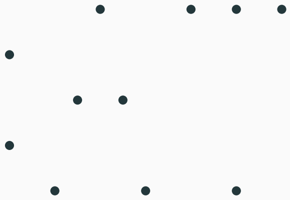
All equivalent!




Representable functor
Universal property
Limit
Initial object
Adjunction
Kan extension
Monad
End
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160

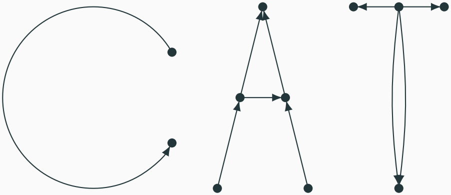
Categories, functors and natural transformations

Categories



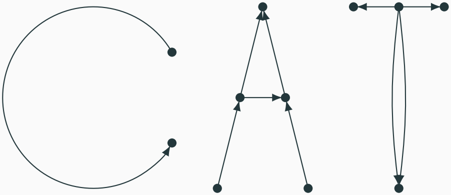
$\mathrm{Ob}\,C : \mathrm{Set}$

Categories



$\mathrm{Ob}\,\mathsf{C} : \mathsf{Set} \qquad C(a, b) = \mathrm{Hom}(a, b) : \mathsf{Set}$

Categories

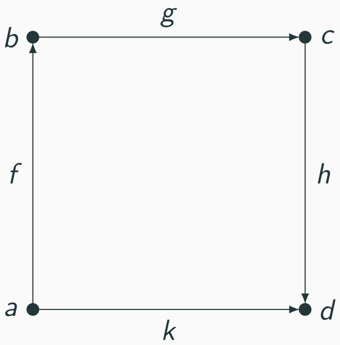


$\mathrm{Ob}\,\mathsf{C} : \mathrm{Set} \qquad C(a, b) = \mathrm{Hom}(a, b) : \mathrm{Set}$

$\circ : C(b, c) \times C(a, b) \to C(a, c) \quad 1_a \in C(a, a)$

$1_b \circ f = f = f \circ 1_a \quad h \circ (g \circ f) = (h \circ g) \circ f$

Commutative diagrams



$$h \circ g \circ f = k$$

Opposite category

C^{op} has the same objects as C

$$C^{\text{op}}(a, b) = C(b, a)$$

Just flip the arrows.

Networks

Categories – macroscopic view

Set: sets and functions

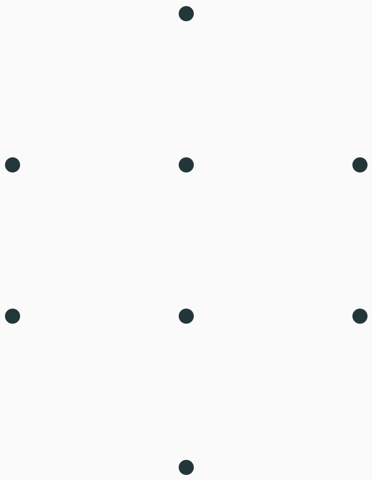
Type: types and computable functions

Vect: vector spaces and linear maps

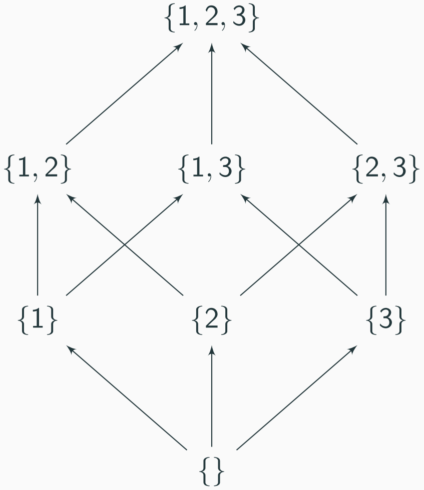
Mon: monoids and monoid maps

Categories – microscopic view

A category is an algebraic structure in its own right.



A set is a category with no arrows (except for identities).



A preorder is a category with at most one arrow between any two objects.

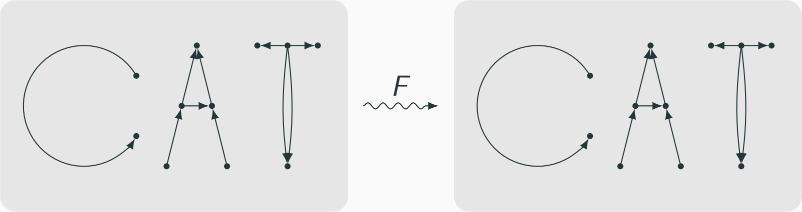
Write

$$a \leq b \text{ if } \operatorname{Hom}(a, b) \text{ is not empty.}$$

Functors

Functors

A functor from $C \to D$ draws a picture of C in D.



$$F : \text{Ob}\, C \to \text{Ob}\, D$$

$$F : C(a, b) \to D(Fa, Fb) \text{ i.e. fmap}$$

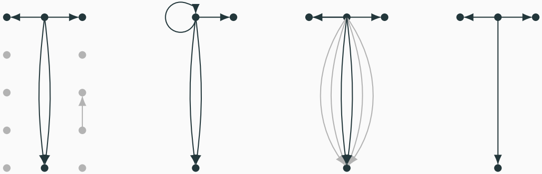
$$F(1) = 1$$

$$F(f \circ g) = F(f) \circ F(g)$$

Functors

Doesn't have to be an exact copy. It could

- miss some objects (not **surjective on objects**)
- collapse some objects (not **injective on objects**)
- miss some arrows in a homset (not **full**)
- collapse some arrows in a homset (not **faithful**)



```haskell
class Functor (f :: * → *) where
  fmap :: (a→b) → f a → f b
```

Presheaf

$$F : C^{\mathsf{op}} \to \mathsf{Set}$$

Contravariant functors

A functor from $C^{\text{op}} \to D$ is called a **contravariant** functor.

$$F(f \circ g) = F(g) \circ F(f)$$

Profunctors

$$C^{\mathrm{op}} \times C \to \mathrm{Set}$$

$$\mathrm{Hom} : C^{\mathrm{op}} \times C \to \mathrm{Set}$$

$$C(-, -) : C^{\mathrm{op}} \times C \to \mathrm{Set}$$

$$a \longrightarrow b$$

$$\mathrm{Hom}(a, b)$$

$$a \longrightarrow b \overset{f}{\longrule} b'$$

$$\mathrm{Hom}(a, b) \longrightarrow \mathrm{Hom}(a, b')$$

$$a \longrightarrow b \overset{f}{\text{———}} b'$$

$$\mathrm{Hom}(a, b) \xrightarrow{(f \circ -)} \mathrm{Hom}(a, b')$$

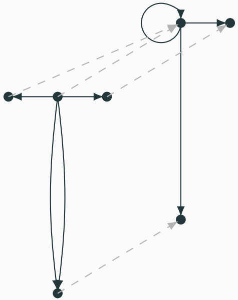
$$\mathrm{Hom}(1, f) := (f \circ -)$$

$$\text{a'} \xrightarrow{\;f\;} \text{a} \longrightarrow \text{b}$$

$$\text{Hom}(a, b) \xrightarrow{(-\circ g)} \text{Hom}(a', b)$$

$$\text{Hom}(g, 1) := (- \circ g)$$

Natural transformations

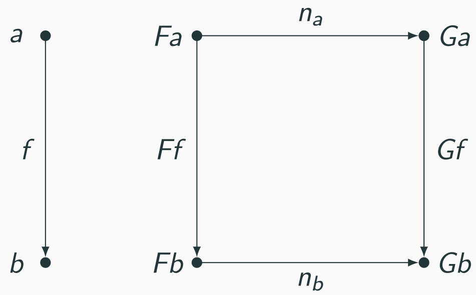


A natural transformation morphs one picture of C into another.

Natural transformations

We have to move every object, but we need to respect the morphisms.

Following an arrow, then translating, should be the same as translating, then following.

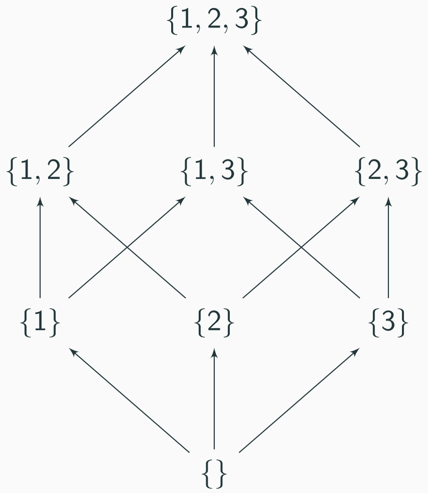


This condition is called **naturality**.

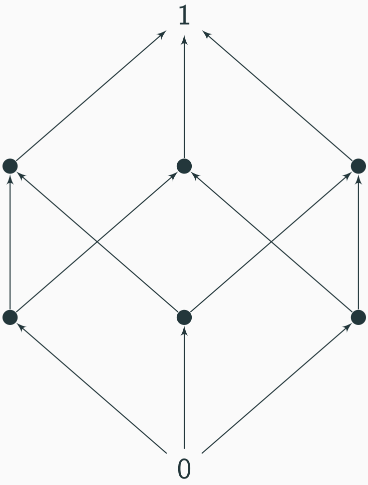
In Haskell we approximate naturality with polymorphism (a stricter condition).

```haskell
type f ⇝ g = ∀ a. f a → g a
```

Initial and terminal objects



{1, 2, 3}
{1, 2}
{1, 3}
{2, 3}
{1}
{2}
{3}
{}


1
0

```haskell
data Void

absurd :: Void → a
absurd v = case v of {}
```

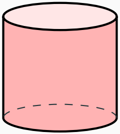
```
unique :: a → ()
unique _ = ()
```

Universal properties

3×4

$X \cap Y$

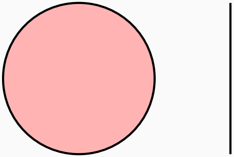
(a, b)

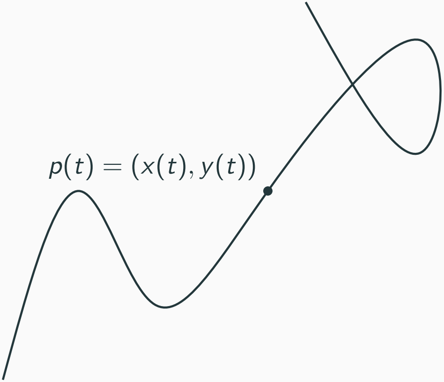


$3 + 4$

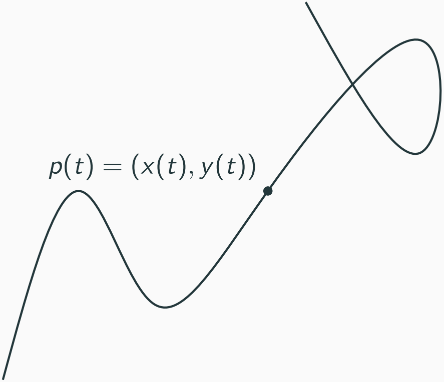
$X \cup Y$

Either a b





$$p : \mathbb{R} \to \mathbb{R} \times \mathbb{R}$$
$$x, y : \mathbb{R} \to \mathbb{R}$$



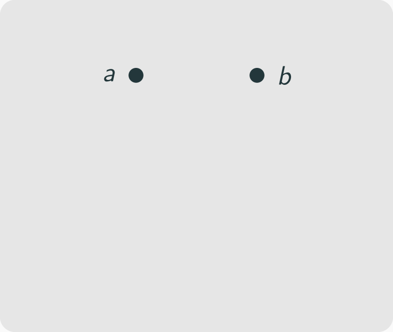
$$p : \mathbb{R} \to \mathbb{R} \times \mathbb{R}$$

$$x, y : \mathbb{R} \to \mathbb{R}$$

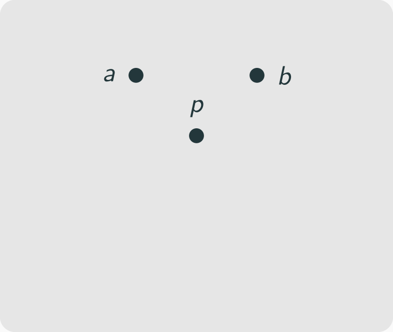
$$\mathrm{Hom}(\mathbb{R}, \mathbb{R}) \times \mathrm{Hom}(\mathbb{R}, \mathbb{R}) \cong \mathrm{Hom}(\mathbb{R}, \mathbb{R} \times \mathbb{R})$$

$$\mathrm{Hom}(x, a) \times \mathrm{Hom}(x, b) \cong \mathrm{Hom}(x, a \times b)$$

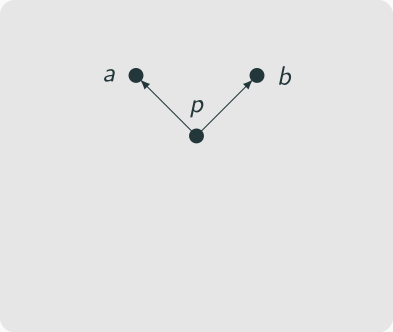
$$\mathrm{Hom}(a, x) \times \mathrm{Hom}(b, x) \cong \mathrm{Hom}(a + b, x)$$



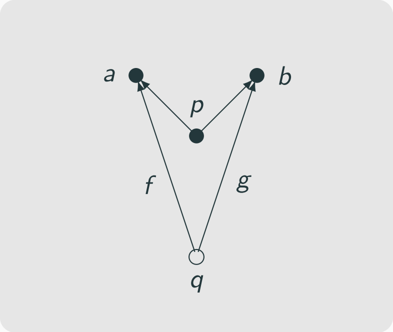
A product of a and b



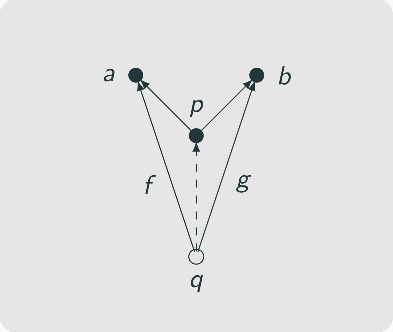
A product of a and b is an object p



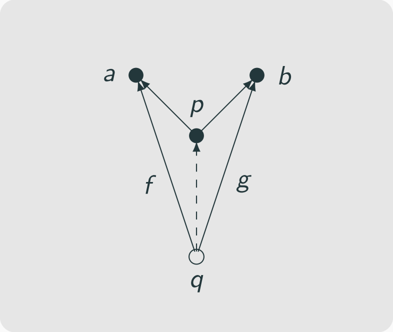
A product of a and b is an object p
with arrows ("projections") to a and b



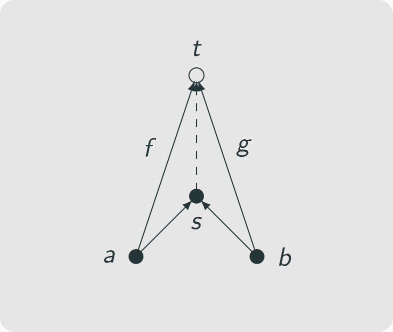
A product of a and b is an object p
with arrows ("projections") to a and b
such that **for any** object q with arrows f, g to a and b



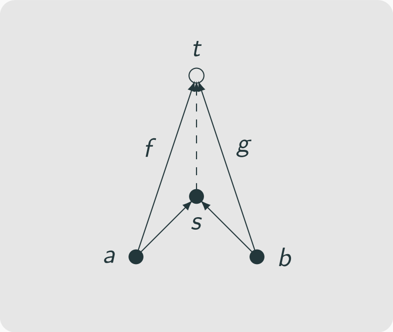
A product of a and b is an object p
with arrows (“projections”) to a and b
such that **for any** object q with arrows f, g to a and b
there exists a **unique** arrow from q to p, which has the property
that composing it with the projections gives back f and g.



$$\mathrm{Hom}(q, a) \times \mathrm{Hom}(q, b) \cong \mathrm{Hom}(q, a \times b)$$



A coproduct of a and b is an object s
with arrows ("injections") from a and b
such that **for any** object t with arrows f, g from a and b
there exists a **unique** arrow from s to t, which has the property
that composing it with the injections gives back f and g.



$$\mathrm{Hom}(a, t) \times \mathrm{Hom}(b, t) \cong \mathrm{Hom}(a + b, t)$$

Products in Type

```
(&&&) :: (a → x) → (a → y) → a → (x, y)
(&&&) f g a = (f a, g a)

split :: (a → (x, y)) → (a → x, a → y)
split f = (fst . f, snd . f)
```

Coproducts in Type

```
to :: (x→a) → (y→a) → Either x y → a
to f g e = case e of
  Left x  → f x
  Right y → g x

from :: (Either x y → a) → (x→a,y→a)
from h = (h.Left,h.Right)
```

Representable functors

$$\mathrm{Hom}(a, x) \times \mathrm{Hom}(b, x) \cong \mathrm{Hom}(a + b, x)$$

$$\mathrm{Hom}(x, a) \times \mathrm{Hom}(x, b) \cong \mathrm{Hom}(x, a \times b)$$

These are isomorphisms of Set-valued functors

A functor isomorphic to $\mathrm{Hom}(-, r)$ or $\mathrm{Hom}(r, -)$ is called **representable**.

We say it is **represented by** r.

$$\mathrm{Hom}(x, a) \times \mathrm{Hom}(x, b) \cong \mathrm{Hom}(x, a \times b)$$

$$\mathrm{Hom}(a, x) \times \mathrm{Hom}(b, x) \cong \mathrm{Hom}(a + b, x)$$

No sums $\Rightarrow$ no choices to be made.

A representable functor $\approx$ a structure with one fixed shape.

Streams

```haskell
data Stream a = a :< Stream a
```

A stream is represented by the natural numbers.

$$\mathbb{N} = 0, 1, 2, 3, \ldots$$

$$\text{Stream } a \cong \text{Hom}(\mathbb{N}, a)$$

```haskell
to :: Stream a → (Natural→a)
to s n = s !! n

from :: (Natural→a) → Stream a
from k = go 0
  where
    go i = k i :< go (i+1)
```

```haskell
to :: Stream a → (Natural→a)
to s n = s !! n

from :: (Natural→a) → Stream a
from k = go 0
  where
    go i = k i :< go (i+1)
```

Apply from to id and you get

$$0, 1, 2, 3, 4, ...$$

We call this the **universal element**.

It's the archetypal stream: the indexing type **reified** as data.

“represented by” $=$ *“indexed by”*

Tuples

A tuple where both parts have the same type is representable.

$$(a, a) \cong \mathrm{Hom}\,(\mathrm{Choice}, a)$$

```
data Choice = L | R
```

$$(L, R)$$

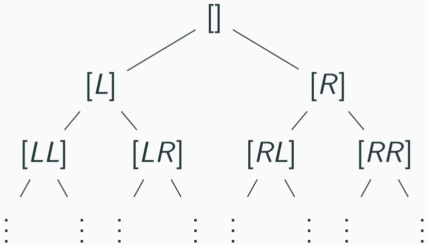
Binary streams

```haskell
data Bin a = Bin a (Bin a) (Bin a)
```

A binary stream is represented by a list of left or right choices.

```haskell
data Choice = L|R
```

```haskell
type Path = [Choice]
```



Yoneda

$$\mathrm{Hom}\left(\mathrm{Hom}(r, -), F\right) \cong F(r) \qquad F : C \to \mathrm{Set}$$

$$\mathrm{Hom}\left(\mathrm{Hom}(-, r), F\right) \cong F(r) \qquad F : C^{\mathrm{op}} \to \mathrm{Set}$$

Yoneda

$$\mathrm{Hom}\left(\mathrm{Hom}(r, -), F\right) \cong F(r) \qquad F : C \to \mathrm{Set}$$

$$\mathrm{Hom}\left(\mathrm{Hom}(-, r), F\right) \cong F(r) \qquad F : C^{\mathrm{op}} \to \mathrm{Set}$$

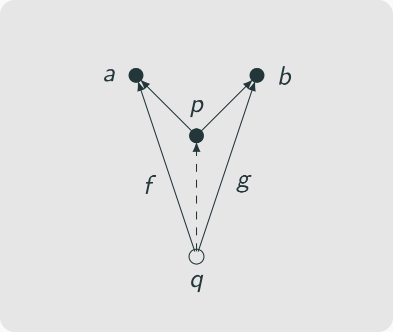
Why?

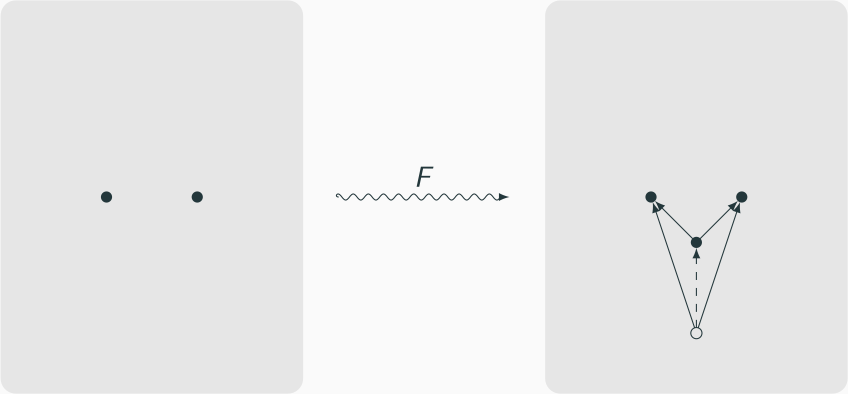
It must send $1_r : r \to r$ to an element of $F(r)$.

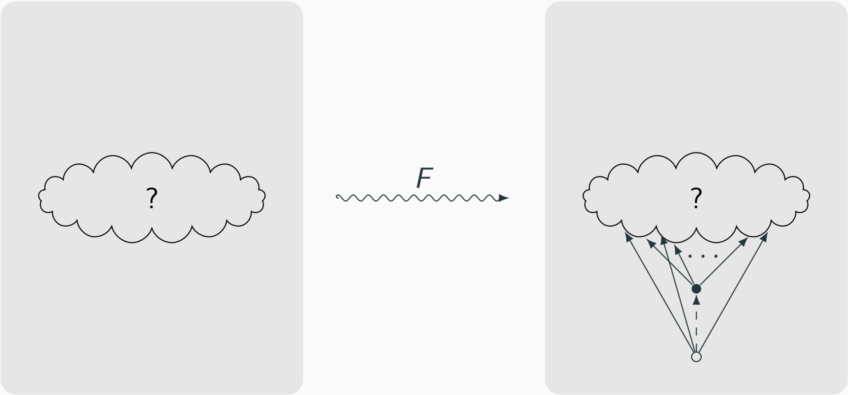
Naturality means there are no more choices to make.

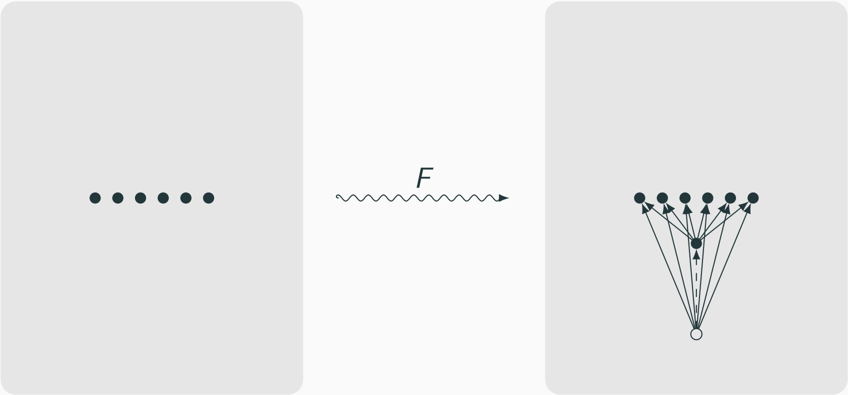
Each element of $F(r)$ defines a natural transformation.

Limits and colimits

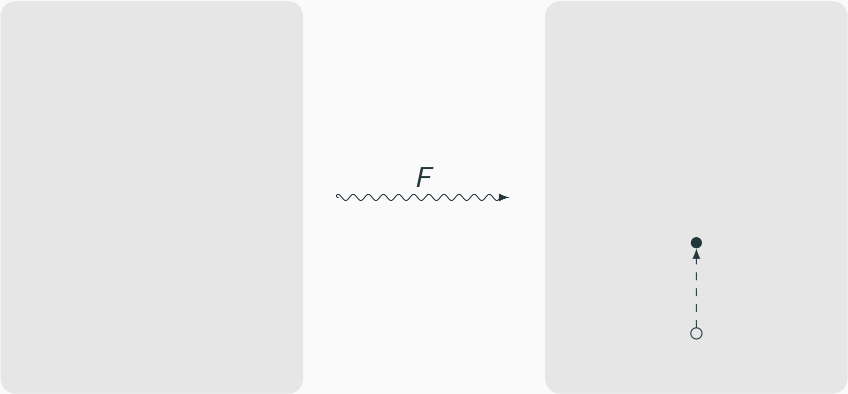

a
b
p
f
g
q


F

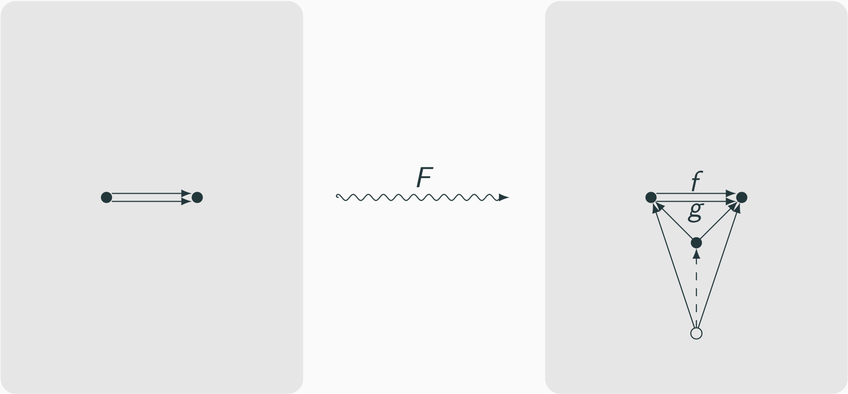

?
F
?


F

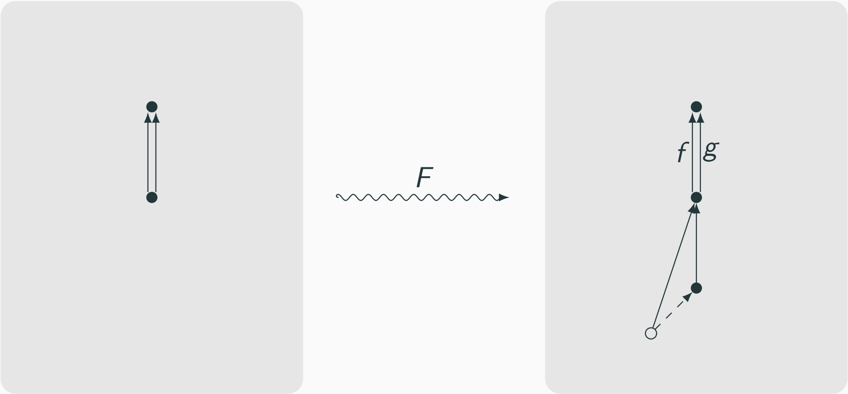
Terminal object



Equalizer



Equalizer



Equalizer

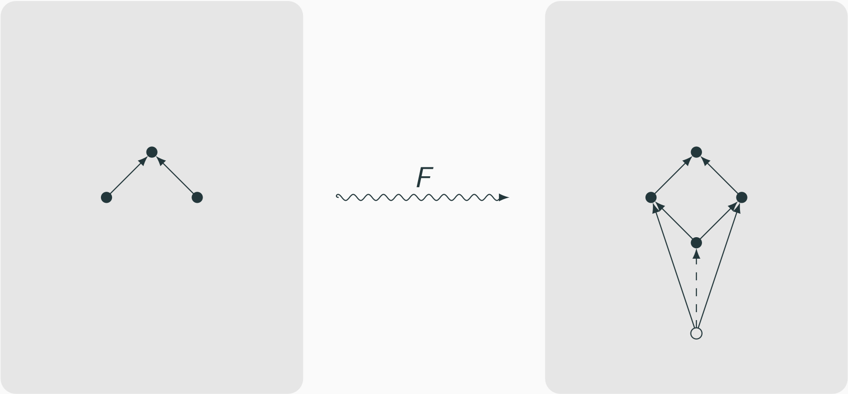
In **Set**

$$\{x \mid f(x) = g(x)\}$$

In **Type**

```
type Equalizer f g x = (x, f x = g x)
```

Pullbacks



Pullbacks

In Set

$$\{(x, y) \mid f(x) = g(y)\}$$

e.g. $f^{-1}(Y) = \{(x, y) \mid f(x) = y\}$

In Type

```
type Pullback f g x = (x, y, f x = g y)
```

Colimits

Flip everything.

Coequalizer

In **Set**

$X/f(a) \sim g(a)$ for all a

Pushout

In **Set**

$X \sqcup Y / f(a) \sim g(a)$ for all a

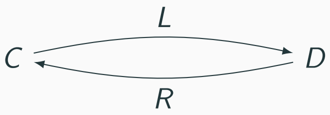
A limit is a generalised product where the projections must satisfy some compatibility conditions

A colimit is a generalised sum where the injections are forced to agree

If we have equalizers and products,
we can build all limits

If we have coequalizers and coproducts,
we can build all colimits

Adjunctions



$$D(Lc, d) \cong C(c, Rd)$$

$$D(Lc, d) \cong C(c, Rd)$$

$$\mathsf{Vect}\left(F\{i, j\}, \mathbb{R}^3\right) \cong \mathsf{Set}\left(\{i, j\}, U\mathbb{R}^3\right)$$

$$\begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} \longleftrightarrow \left(i \mapsto \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, j \mapsto \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix} \right)$$

$$D(Lc, d) \cong C(c, Rd)$$

$$\text{Type}\,(a \times b, c) \cong \text{Type}\,(a, b \to c)$$

```
curry :: ((a, b) → c) → a → b → c
uncurry :: (a → b → c) → (a, b) → c
```

Right adjoints preserve limits

Left adjoints preserve colimits

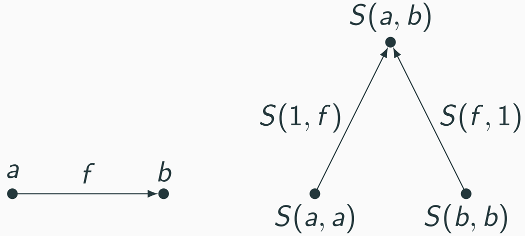
$$
\begin{aligned}
U1 &\cong 1 \\
U(a \times b) &\cong U(a) \times U(b)
\end{aligned}
$$

$$
\begin{aligned}
F0 &\cong 0 \\
F(a + b) &\cong F(a) + F(b)
\end{aligned}
$$

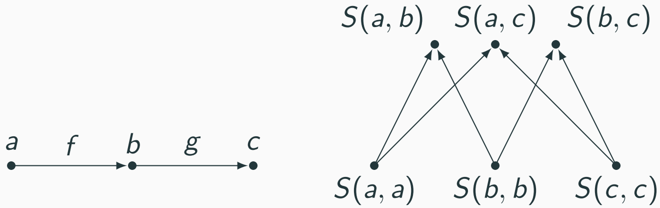
Monads and comonads

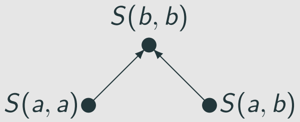
Ends and coends

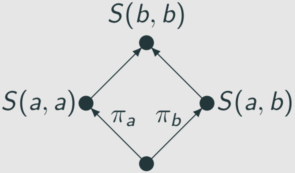
Limits for profunctors?

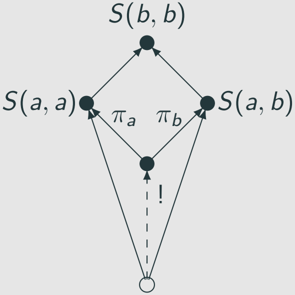


a
f
b
S(a, b)
S(1, f)
S(f, 1)
S(a, a)
S(b, b)


a
f
b
g
c
S(a, b)
S(a, c)
S(b, c)
S(a, a)
S(b, b)
S(c, c)


S(b, b)
S(a, a)
S(a, b)


$S(b, b)$
$S(a, a)$
$S(a, b)$
π_a
π_b



$$\pi_a : \int_c S(c, c) \to S(a, a)$$

$$\int_a S(a,a) \approx \text{ data End s} = \text{End } (\forall \text{a. s a a})$$

$$\int^a S(a,a) \approx \text{ data Coend s} = \forall \text{a. Coend (s a a)}$$

```
proj :: End s → s b b
proj (End x) = x

inj :: s b b → Coend s
inj x = Coend x
```

(Co)end calculus

$$\int_a \int_b S(a,a,b,b) \cong \int_b \int_a S(a,a,b,b) \cong \int_{(a,b)} S(a,a,b,b)$$

$$C\left(\int^a S(a,a), b\right) \cong \int_a C\left(S(a,a), b\right)$$

$$C\left(b, \int_a S(a,a)\right) \cong \int_a C\left(b, S(a,a)\right)$$

$$F \cong \int_c \mathrm{Hom}\left(C(c,-), Fc\right) \cong \int^c F(c) \times C(c,-)$$

Natural transformations are ends

$$\mathsf{Nat}(F, G) = \int_s \mathsf{Hom}(Fs, Gs)$$

Let's prove that `a → f b` and `r ⇝ f` are isomorphic types, where `type r x = (a, b → x)`.

$$
\begin{aligned}
&\mathrm{Set}(a, Fb) \\
&\cong \mathrm{Set}(a, \int_x \mathrm{Set}(\mathrm{Set}(b, x), Fx)) \text{ (Yoneda)} \\
&\cong \int_x \mathrm{Set}(a, \mathrm{Set}(\mathrm{Set}(b, x), Fx)) \text{ (end preserves homsets)} \\
&\cong \int_x \mathrm{Set}(a \times \mathrm{Set}(b, x), Fx) \text{ (uncurry)} \\
&\cong \int_x \mathrm{Set}((a \times \mathrm{Set}(b, -))x, Fx) \text{ (extract functor)} \\
&\cong \mathrm{Nat}((a \times \mathrm{Set}(b, -)), F) \text{ (natural transformations as ends)}
\end{aligned}
$$

We can use this to show that `(a,b → c)` is isomomorphic to
`∀ f. Functor f ⇒(a → f b) → f c`

$$\int_F \mathrm{Set}(\mathcal{F}(G, F), HF) \cong HG \text{ (Yoneda lemma)}$$

$$\int_F \mathrm{Set}(\mathcal{F}(G, F), Fc) \cong Gx \text{ (choose } H = -(c) \text{)}$$

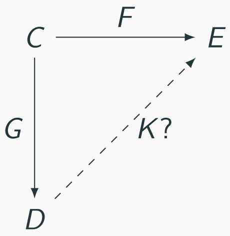
$$\int_F \mathrm{Set}(\mathcal{F}(a \times \mathrm{Set}(b, -), F), Fc) \cong (a, \mathrm{Set}(b, c)) \text{ (set } G = (a \times \mathrm{Set}(b, -)) \text{)}$$

$$\begin{aligned} &\int_F \mathrm{Set}(\mathrm{Set}(a, Fb), Fc) \\ &\cong \int_F \mathrm{Set}(\mathcal{F}((a \times \mathrm{Set}(b, -)), F), Fc) \text{ (last slide)} \\ &\cong (a \times \mathrm{Set}(b, c)) \text{ (line above = Yoneda)} \end{aligned}$$

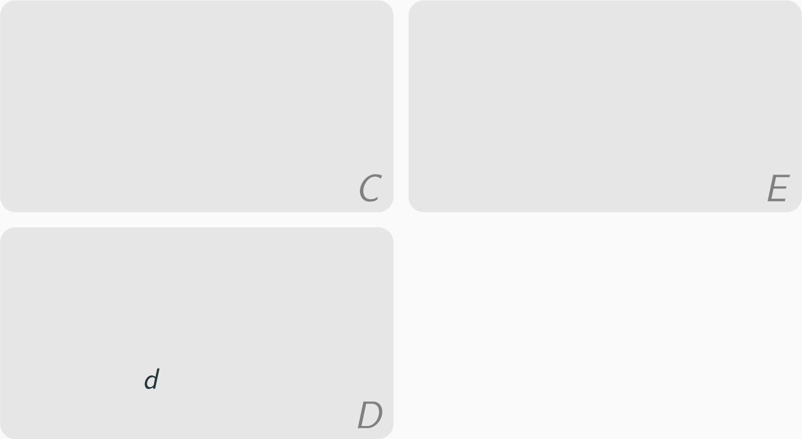
Lenses

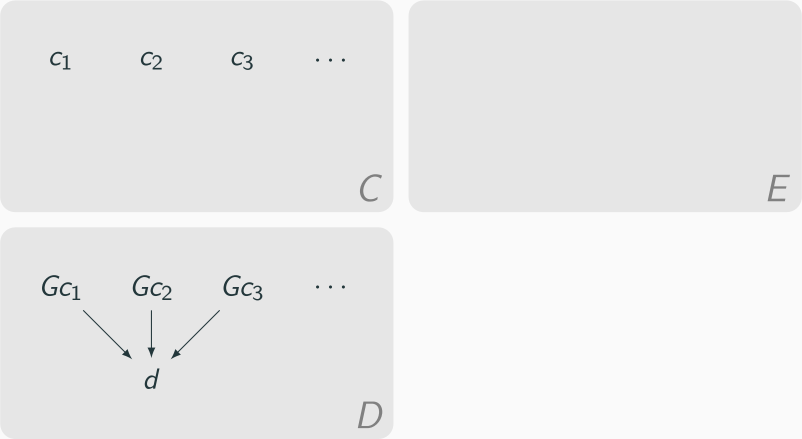
$$(s \to a, a \to s \to s) \cong \forall f.\ \text{Functor}\ f \Rightarrow (a \to f\ a) \to s \to f\ s$$

Kan extensions


F
C
E
G
K?
D

Lan, Ran


C
E
d
D

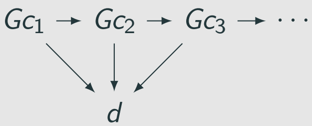


c_1 c_2 c_3 $\cdots$
C
E
Gc_1 Gc_2 Gc_3 $\cdots$
d
D

$$c_1 \longrightarrow c_2 \longrightarrow c_3 \longrightarrow \cdots$$

C

E



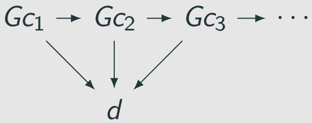
D

$$c_1 \longrightarrow c_2 \longrightarrow c_3 \longrightarrow \cdots$$

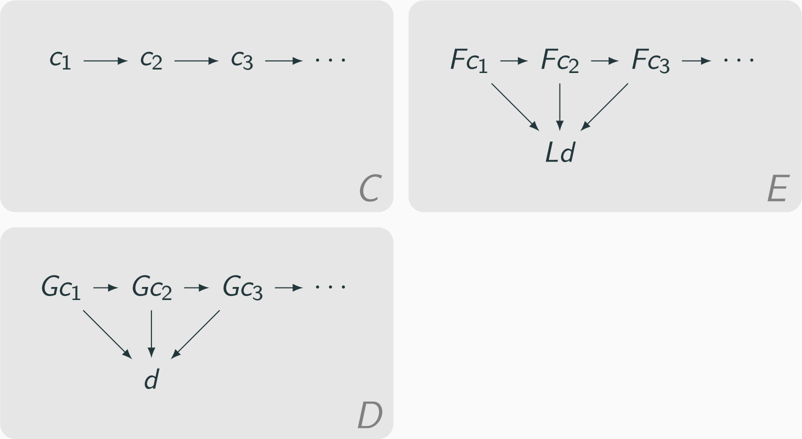
C

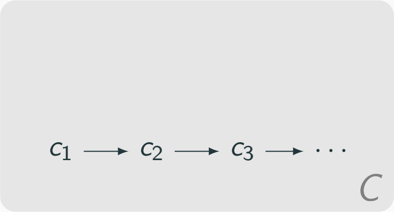
$$Fc_1 \to Fc_2 \to Fc_3 \to \cdots$$

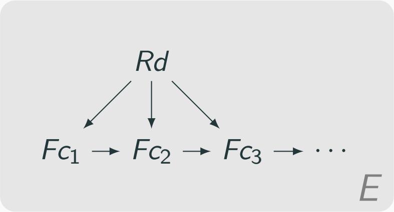
E

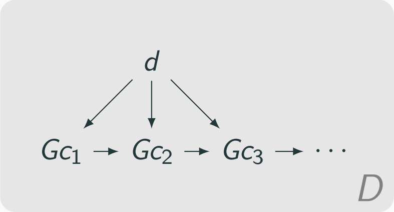


D


c1 → c2 → c3 → ···
C
Fc1 → Fc2 → Fc3 → ···
Ld
E
Gc1 → Gc2 → Gc3 → ···
d
D


$c_1 \to c_2 \to c_3 \to \cdots$
C


Rd
$Fc_1 \to Fc_2 \to Fc_3 \to \cdots$
E


d
$Gc_1 \to Gc_2 \to Gc_3 \to \cdots$
D

$$(\mathrm{Lan}_G\, F)(d) = \int^{c} D(Gc, d).Fc$$

$$(\mathrm{Ran}_G\, F)(d) = \int_{c} \mathrm{Hom}(D(d, Gc), Fc)$$

```
data Lan g f d = ∀ c. Lan (g c → d) (f c)

newtype Ran g f d = Ran (∀ c. (d → g c) → f c)
```

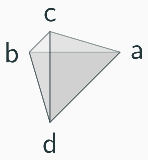
The original Kan extension: geometric realisation

$$\{a, b, c\}\{a, b, d\}\{a, c, d\}\{b, c, d\}$$

$$\{a, b\}\{a, c\}\{a, d\}\{b, c\}\{b, d\}\{c, d\}$$

$$\{a\}\{b\}\{c\}\{d\}$$

$\{a, b, c, d\} \longrightarrow$



Day convolution

$F, G : C \to \mathrm{Set}$

$F \boxtimes G : C \times C \to \mathrm{Set}$

$(F \boxtimes G)(c_1, c_2) := F(c_1) \times G(c_2)$

Day convolution

$F, G : C \to \mathsf{Set}$

$F \boxtimes G : C \times C \to \mathsf{Set}$

$(F \boxtimes G)(c_1, c_2) := F(c_1) \times G(c_2)$

$$\times : C \times C \to C$$

$$F \otimes G := \mathsf{Lan}_{\times} \boxtimes$$

```
data Day f g c
= ∀ c1 c2. Day (c1 → c2 → c) (f c1) (g c2)
```

Summary

Themes

- Arrows are more important than objects
- Duality
- Weakening adds structure
- Understanding is hard but proofs are easy c.f. number theory

Where to from here?

- Enriched categories
- Higher categories
- Topos theory
- ...

Further reading/watching

- Eugenia Cheng, The Catsters (YouTube)
- Bartosz Milewski (videos and blogposts)
- comonad.com (Ed Kmett and Dan Doel)
- Emily Riehl, Category Theory in Context
- Tom Leinster, Basic Category Theory
- David Spivak, Category theory for the sciences
- Lawvere and Schanuel, Conceptual mathematics
- the nLab

References

- Categories for the working mathematician
- This is the (co)end, my only (co)friend.
- A representation theorem for second order functionals.
- Notions of computation as monoids.