

$$\begin{array}{ccc} \text{Hom}(A, A) & \xrightarrow{\text{Hom}(A, f)} & \text{Hom}(A, X) \\ \downarrow \Phi_A & & \downarrow \Phi_X \\ F(A) & \xrightarrow{Ff} & F(X) \end{array}$$

$\begin{array}{ccc} \text{id}_A \vdash & \longrightarrow & f \\ \downarrow & & \downarrow \\ u \vdash & \longrightarrow & (Ff)u = \Phi_X(f) \end{array}$

## What you needa know about Yoneda

Emma Bach (she/her)

Seminar on Functional Programming and Logic, Summer Semester 2025

# Motivation

- ▶ A common sentiment in many cultures is the idea that people are defined by how they interact with their surroundings.

---

<sup>1</sup>Quoted as a proverb in *Don Quixote*

# Motivation

- ▶ A common sentiment in many cultures is the idea that people are defined by how they interact with their surroundings.
- ▶ “*Tell me your company, and I will tell you what you are.*”<sup>1</sup>

---

<sup>1</sup>Quoted as a proverb in *Don Quixote*

# Motivation

- ▶ A common sentiment in many cultures is the idea that people are defined by how they interact with their surroundings.
- ▶ “*Tell me your company, and I will tell you what you are.*”<sup>1</sup>
- ▶ The Yoneda Lemma is the result of applying this way of thinking to mathematical objects within the extremely general setting of *category theory*.

---

<sup>1</sup>Quoted as a proverb in *Don Quixote*

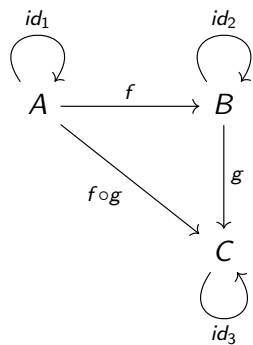
# Motivation

- ▶ A common sentiment in many cultures is the idea that people are defined by how they interact with their surroundings.
- ▶ “*Tell me your company, and I will tell you what you are.*”<sup>1</sup>
- ▶ The Yoneda Lemma is the result of applying this way of thinking to mathematical objects within the extremely general setting of *category theory*.
- ▶ As a result, a category  $\mathbb{C}$  is often best understood by instead studying functors from that category into  $\mathbf{Set}$ .

---

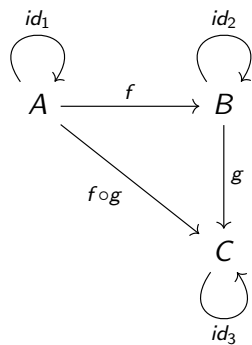
<sup>1</sup>Quoted as a proverb in *Don Quixote*

# Categories



A category  $\mathbb{C}$  consists of:

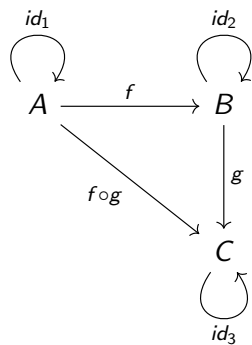
# Categories



A category  $\mathbb{C}$  consists of:

- ▶ a collection  $|\mathbb{C}|$  of *objects*;

# Categories

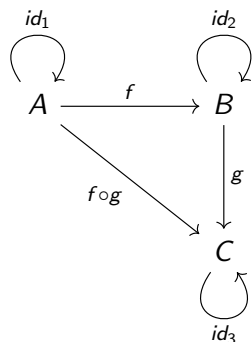


A category  $\mathbb{C}$  consists of:

- ▶ a collection  $|\mathbb{C}|$  of *objects*;
- ▶ for all  $A, B \in |\mathbb{C}|$ , a collection  $\mathbb{C}(A, B)$  of *morphisms* from  $A$  to  $B$ ;



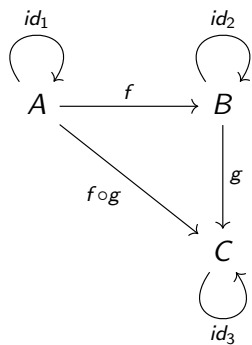
# Categories



A category  $\mathbb{C}$  consists of:

- ▶ a collection  $|\mathbb{C}|$  of *objects*;
- ▶ for all  $A, B \in |\mathbb{C}|$ , a collection  $\mathbb{C}(A, B)$  of *morphisms* from  $A$  to  $B$ ;
- ▶ for all  $A \in |\mathbb{C}|$ , an *identity morphism*  $id_A \in \mathbb{C}(A, A)$ ;

# Categories

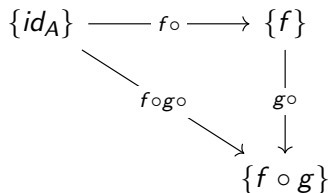
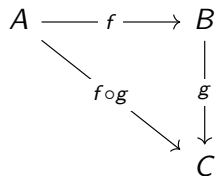


A *category*  $\mathbb{C}$  consists of:

- ▶ a collection  $|\mathbb{C}|$  of *objects*;
- ▶ for all  $A, B \in |\mathbb{C}|$ , a collection  $\mathbb{C}(A, B)$  of *morphisms* from  $A$  to  $B$ ;
- ▶ for all  $A \in |\mathbb{C}|$ , an *identity morphism*  $id_A \in \mathbb{C}(A, A)$ ;
- ▶ an associative *composition morphism*  $f \circ g \in \mathbb{C}(A, C)$  for each pair of morphisms  $f \in \mathbb{C}(A, B)$ ,  $g \in \mathbb{C}(B, C)$ .

If  $\mathbb{C}(A, B)$  is a set, we call it the *homset* from  $A$  to  $B$ .

# Homfunctors



- ▶ For any category  $\mathbb{C}$ , a homset  $\mathbb{C}(A, B)$  is a set of morphisms.
- ▶ We define a functor  $\mathbb{C}(A, -) : \mathbb{C} \rightarrow \mathbf{Set}$ :
  - ▶  $\mathbb{C}(A, -)$  maps an Object  $B$  to the Homset  $\mathbb{C}(A, B)$
  - ▶ A morphism  $f : \mathbb{C}(B, C)$  is mapped to the morphism  $f \circ : \mathbb{C}(A, B) \rightarrow \mathbb{C}(A, C)$

# Natural Transformations

- ▶ A structure-preserving map between functors.
  - ▶ Let  $F, G : \mathbb{C} \rightarrow \mathbb{D}$  be functors.
  - ▶ A natural transformation  $\phi$  is an indexed family of morphisms  $\phi_A \in \mathbb{D}(F(A), G(A))$  from  $F(A)$  to  $G(A)$
  - ▶ These morphisms satisfy the following *naturality condition*:

$$\forall f \in \mathbb{C}(A, B) : \phi_B \circ F(f) = G(f) \circ \phi_A$$

- ▶ Given two functors  $F$  and  $G$ , we write the collection of all natural transformation between them as  $\text{Nat}(F, G)$ .

# Naturality from Polymorphism

- The naturality condition resembles an equality we saw a few weeks ago:

$$r_B \circ \text{map}(a) = \text{map}(a) \circ r_A$$

# Naturality from Polymorphism

- ▶ The naturality condition resembles an equality we saw a few weeks ago:

$$r_B \circ \text{map}(a) = \text{map}(a) \circ r_A$$

- ▶ This is the free theorem we got for a parametrically polymorphic function  $r : [X] \rightarrow [X]$  and an arbitrary function  $a : A \rightarrow B$ .

# Naturality from Polymorphism

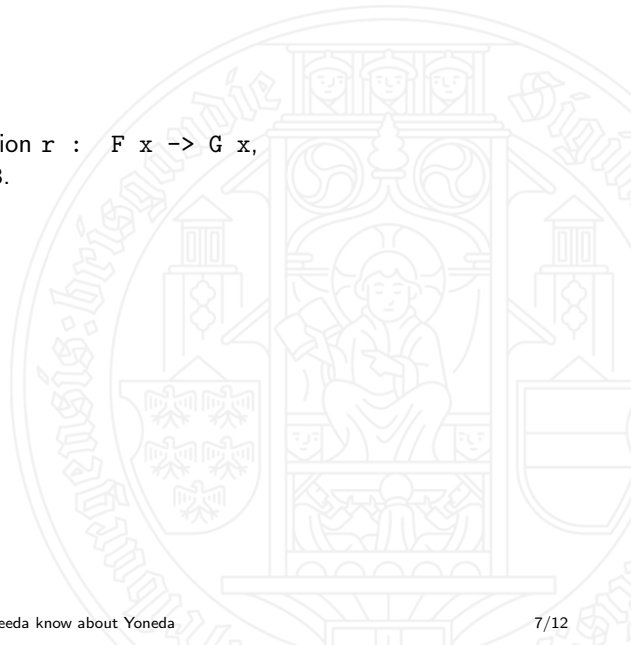
- ▶ The naturality condition resembles an equality we saw a few weeks ago:

$$r_B \circ \text{map}(a) = \text{map}(a) \circ r_A$$

- ▶ This is the free theorem we got for a parametrically polymorphic function  $r :: [X] \rightarrow [X]$  and an arbitrary function  $a :: A \rightarrow B$ .
- ▶ This free theorem proves that  $r$  is a natural transformation from the list functor to itself.

# Naturality from Polymorphism

- ▶ In general, assume we have:
  - ▶ two functors  $F$  and  $G$ ,
  - ▶ a parametrically polymorphic function  $r : F\ x \rightarrow G\ x$ ,
  - ▶ an arbitrary function  $f : A \rightarrow B$ .





# Naturality from Polymorphism

- ▶ In general, assume we have:
  - ▶ two functors  $F$  and  $G$ ,
  - ▶ a parametrically polymorphic function  $r : F\ x \rightarrow G\ x$ ,
  - ▶ an arbitrary function  $f : A \rightarrow B$ .
- ▶ Then we get the following free theorem:

$$r \ . \ fmap\ f = fmap\ f \ . \ r$$

# Naturality from Polymorphism

- ▶ In general, assume we have:
  - ▶ two functors  $F$  and  $G$ ,
  - ▶ a parametrically polymorphic function  $r : F\ x \rightarrow G\ x$ ,
  - ▶ an arbitrary function  $f : A \rightarrow B$ .
- ▶ Then we get the following free theorem:

$$r \cdot \text{fmap } f = \text{fmap } f \cdot r$$

- ▶ In categorical notation:

$$r_B \circ F(f) = G(f) \circ r_A$$

# Naturality from Polymorphism

- ▶ In general, assume we have:
  - ▶ two functors  $F$  and  $G$ ,
  - ▶ a parametrically polymorphic function  $r : F\ x \rightarrow G\ x$ ,
  - ▶ an arbitrary function  $f : A \rightarrow B$ .
- ▶ Then we get the following free theorem:

$$r \cdot \text{fmap } f = \text{fmap } f \cdot r$$

- ▶ In categorical notation:

$$r_B \circ F(f) = G(f) \circ r_A$$

- ▶ So our free theorem is a proof that any parametrically polymorphic function  $r$  is a natural transformation!

# Naturality from Polymorphism

- ▶ In general, assume we have:
  - ▶ two functors  $F$  and  $G$ ,
  - ▶ a parametrically polymorphic function  $r : F\ x \rightarrow G\ x$ ,
  - ▶ an arbitrary function  $f : A \rightarrow B$ .
- ▶ Then we get the following free theorem:

$$r \ . \ \text{fmap } f = \text{fmap } f \ . \ r$$

- ▶ In categorical notation:

$$r_B \circ F(f) = G(f) \circ r_A$$

- ▶ So our free theorem is a proof that any parametrically polymorphic function  $r$  is a natural transformation!
- ▶ It turns out that parametrically polymorphic functions correspond exactly to natural transformations between endofunctors  $\text{Set} \rightarrow \text{Set}$ .

# The Yoneda Lemma

- The Yoneda Lemma states:

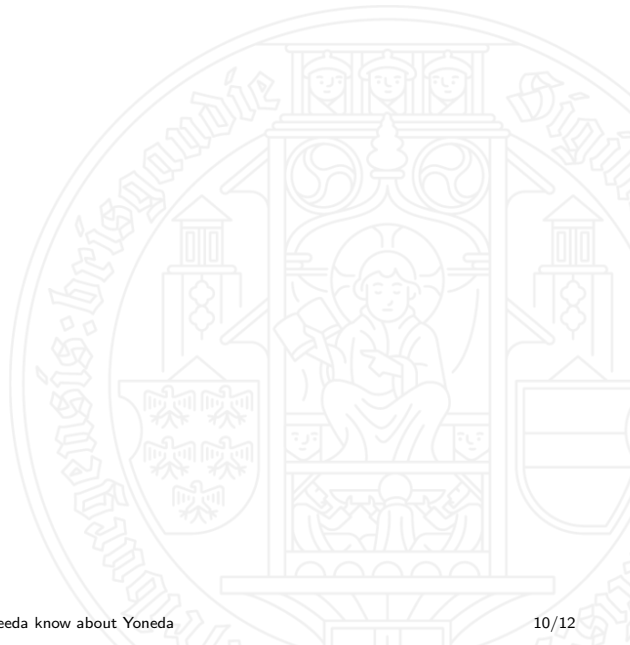
Nat

# The Yoneda Lemma

Proof



...



# Instances of the Yoneda Lemma

- ▶ Cayley's theorem in group theory
- ▶ Countless theorems in algebra, particularly in algebraic topology
- ▶ Proofs by indirect inequality:  $b \preceq a$  iff.  $\forall c : (a \preceq c) \implies (b \preceq c)$
- ▶ Profunctor optics in functional programming



# Profunctor Optics

