universität freiburg

$$
\begin{tikzcd}
\mathrm{Hom}(A, A) \arrow[rrr, "\mathrm{Hom}(A,f)"] \arrow[ddd, "\Phi_A"'] & & & \mathrm{Hom}(A, X) \arrow[ddd, "\Phi_X"] \\
& \mathrm{id}_A \arrow[r, mapsto] \arrow[d, mapsto] & f \arrow[d, mapsto] & \\
& u \arrow[r, mapsto] & (Ff)u = \Phi_X(f) & \\
F(A) \arrow[rrr, "Ff"'] & & & F(X)
\end{tikzcd}
$$

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.

[1]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."*[1]

[1] 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.*"[1]
- The Yoneda Lemma is the result of applying this way of thinking to mathematical objects within the extremely general setting of *category theory*.

[1] 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.*"[1]
- 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 $\mathbb{S}et$.

[1] Quoted as a proverb in *Don Quixote*

Categories

$$
\begin{tikzcd}
A \arrow[loop above, "id_1"] \arrow[r, "f"] \arrow[dr, "f \circ g"'] & B \arrow[loop above, "id_2"] \arrow[d, "g"] \\
& C \arrow[loop below, "id_3"]
\end{tikzcd}
$$

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

Categories

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[dr, "f \circ g"'] \arrow[loop above, "id_1"] & B \arrow[d, "g"] \arrow[loop above, "id_2"] \\ & C \arrow[loop below, "id_3"] \end{tikzcd}$$

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

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

Categories

$$\begin{tikzcd}
A \arrow[loop above, "id_1"] \arrow[r, "f"] \arrow[dr, "f \circ g"'] & B \arrow[loop above, "id_2"] \arrow[d, "g"] \\
& C \arrow[loop below, "id_3"]
\end{tikzcd}$$

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

$$\begin{tikzcd}
A \arrow[loop above, "id_1"] \arrow[r, "f"] \arrow[dr, "f \circ g"'] & B \arrow[loop above, "id_2"] \arrow[d, "g"] \\
& C \arrow[loop below, "id_3"]
\end{tikzcd}$$

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

$$\begin{tikzcd}
A \arrow[loop above, "id_1"] \arrow[r, "f"] \arrow[dr, "f \circ g"'] & B \arrow[loop above, "id_2"] \arrow[d, "g"] \\
& C \arrow[loop below, "id_3"]
\end{tikzcd}$$

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 $f \in \mathbb{C}(A, B)$, $g \in \mathbb{C}(A, B)$.

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

Homfunctors

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[dr, "f \circ g"'] & B \arrow[d, "g"] \\ & C \end{tikzcd}$$

$$\begin{tikzcd} \{id_A\} \arrow[r, "f\circ"] \arrow[dr, "f \circ g \circ"'] & \{f\} \arrow[d, "g\circ"] \\ & \{f \circ g\} \end{tikzcd}$$

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

Natural Transformations

- A structure-preserving map between functors.
  - Let $F, G : \mathbb{C} \to \mathbb{D}$ be functors.
  - A natural transformation ϕ 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 $\mathrm{Nat}(F, G)$.

Naturality from Polymorphism

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

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

Naturality from Polymorphism

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

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

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

Naturality from Polymorphism

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

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

- This is the free theorem we got for a parametrically polymorphic function `r :: [X] -> [X]` and an arbitrary function `a : A -> 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 -> G x`,
  - an arbitrary function `f :: A -> B`.

Naturality from Polymorphism

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

```
r . fmap f = fmap f . 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 -> G x`,
  - an arbitrary function `f :  A -> B`.
- Then we get the following free theorem:

```
r .  fmap f = 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!

Naturality from Polymorphism

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

```
r . fmap f = 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 $\mathbb{S}et \to \mathbb{S}et$.

The Yoneda Lemma

- The Yoneda Lemma states:

$$\mathrm{Nat}$$

The Yoneda Lemma

Proof

. . .

Instances of the Yoneda Lemma

- Cayley's theorem in group theory
- Countless theorems in algebra, particulary 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