Introduction to Category Theory for Functional Programming

Andrew R. Plummer

Department of Linguistics
The Ohio State University

14 Oct, 2009

OUTLINE

CATEGORIES

CATEGORY AXIOMS (1)

A *category* **C** comprises the following:

- A collection of *objects*, denoted by $Ob(\mathbf{C})$;
- A collection of *arrows*, denoted by $Ar(\mathbf{C})$;

Neither of these collections need be sets.

CATEGORIES

CATEGORY AXIOMS (2)

- Operations dom and cod from the collection of arrows to the collection of objects:

$$\text{dom} : Ar(\mathbf{C}) \to Ob(\mathbf{C}) \quad \text{and} \quad \text{cod} : Ar(\mathbf{C}) \to Ob(\mathbf{C}).$$

- For any arrow $f \in Ar(\mathbf{C})$, dom f is called the *domain* of f, and cod f is called the *codomain* of f. If dom $f = a$ and cod $f = b$, we represent this as

$$f : a \to b \quad \text{or} \quad a \xrightarrow{f} b;$$

CATEGORIES

CATEGORY AXIOMS (3)

An operation $\circ$ that assigns to each pair $\langle g, f \rangle$ of arrows $f, g \in Ar(\mathbf{C})$ that satisfies cod f = dom g, an arrow $g \circ f$, called the *composite* of f and g, such that

$$\operatorname{dom} g \circ f = \operatorname{dom} f, \quad \text{and} \quad \operatorname{cod} g \circ f = \operatorname{cod} g;$$

That is,

$$(g \circ f) : a \to c$$

CATEGORIES

CATEGORY AXIOMS (4)

For all objects $b \in Ob(\mathbf{C})$, there is an arrow $\mathrm{id}_b : b \to b$ in $Ar(\mathbf{C})$, called the *identity arrow on b*, such that the following identity holds for all arrows $f : a \to b$ and $g : b \to c$ in $Ar(\mathbf{C})$:

$$\mathrm{id}_b \circ f = f \quad \text{and} \quad g \circ \mathrm{id}_b = g;$$

That is,

$$(f \circ \mathrm{id}_a) : a \to b \qquad (\mathrm{id}_b \circ g) : b \to c$$
$$(f \circ \mathrm{id}_a) = f \qquad (\mathrm{id}_b \circ g) = g$$

CATEGORIES

CATEGORY AXIOMS (5)

For all arrows $f : a \to b$, $g : b \to c$, and $h : c \to d$ in $Ar(\mathbf{C})$ the following identity holds:

$$h \circ (g \circ f) = (h \circ g) \circ f.$$

MONOIDS

MONOIDS

A monoid is a triple $\mathcal{M} = \langle M, *, e \rangle$ such that

- M is a set,
- $*$ is an associative binary operation on M,
- e is the identity element for $*$ on M.

Monoids are commonly used throughout computer science:

- Transition monoids and syntactic monoids in describing a finite state machine,
- Trace monoids and history monoids for process calculi and concurrent computing.

HASKELL EXAMPLE

The Haskell is a purely functional programming language based on the typed λ-calculus. All of our programming examples are based on Haskell syntax.

LISTS

Let a be a type (e.g., Int). Then $\mathcal{L}_a = \langle [a], (++), [\,] \rangle$ is a monoid.

- ([ ] ++ [1,2,3]) = ([1,2,3] ++ [ ]) = [1,2,3]
- (([1,2] ++ [3,4]) ++ [5,6]) = ([1,2] ++ ([3,4] ++ [5,6])) = [1,2,3,4,5,6]

MONOIDS AS CATEGORIES

MONOID CATEGORY

Given a monoid $\mathcal{M} = \langle M, *, e \rangle$, let $\mathbf{C}_{\mathcal{M}}$ be the following category:

- $Ob(\mathbf{C}_{\mathcal{M}}) = \{0\}$ (any set with one element),
- $Ar(\mathbf{C}_{\mathcal{M}}) = M$,
- $g \circ f = g * f$.
- $\mathrm{id}_0 = e$, for $0 \in Ob(\mathbf{C}_{\mathcal{M}})$.

Since $*$ is associative in $\mathcal{M}$, $\circ$ is associative on $Ar(\mathbf{C}_{\mathcal{M}})$. Since e is the identity in $\mathcal{M}$, e is the identity arrow for $0 \in Ob(\mathbf{C}_{\mathcal{M}})$.

PREORDERS AS CATEGORIES

PREORDERS

A preorder $\mathcal{P}$ is a set P equipped with a relation $\sqsubseteq$ which is reflexive and transitive.

PREORDER CATEGORY

Given a preorder $\mathcal{P} = \langle P, \sqsubseteq \rangle$, let $\mathbf{C}_{\mathcal{P}}$ be the following category:

- $Ob(\mathbf{C}_{\mathcal{P}}) = P$,
- $Ar(\mathbf{C}_{\mathcal{P}}) = \{\langle a, b\rangle : a \to b \mid a \sqsubseteq b\}$,
- $\langle b, c\rangle \circ \langle a, b\rangle = \langle a, c\rangle$.
- $\mathrm{id}_a = \langle a, a\rangle$, for $a \in Ob(\mathbf{C}_{\mathcal{P}})$.

OUTLINE

FUNCTORS

Purely functional programming is founded entirely on category theory. Functors are used to handle side effects, and real world interaction.

FUNCTOR (1)

A *functor F* from a category **C** to a category **D** is a function such that:

> Each object $a \in Ob(\mathbf{C})$ is mapped to an object $F(a) \in Ob(\mathbf{D})$.

FUNCTORS

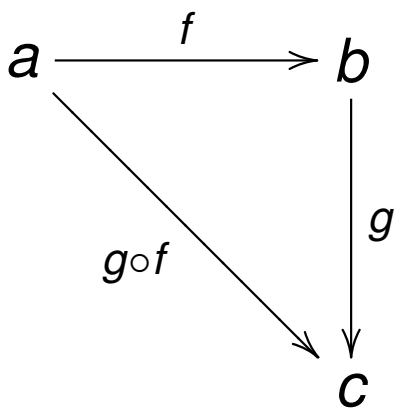
FUNCTOR (2)

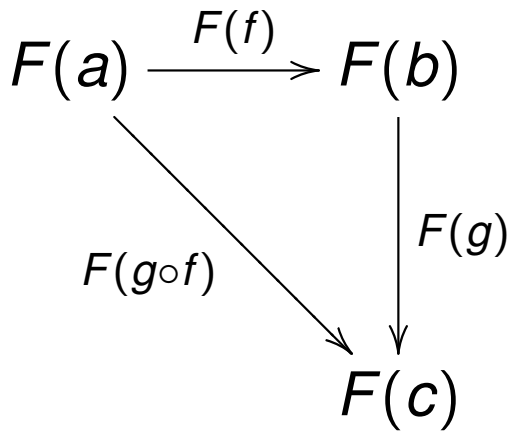
Each arrow $f : a \to b$ in $Ar(\mathbf{C})$ is mapped to an arrow $F(f) : F(a) \to F(b)$ in $Ar(\mathbf{D})$ such that:

- For all objects $a \in Ob(\mathbf{C})$, the identity arrow on a is mapped to the identity arrow on $F(a)$. That is,
$$F(\mathrm{id}_a) = \mathrm{id}_{F(a)}.$$
- For all pairs $\langle g, f \rangle$ where $f, g \in Ar(\mathbf{C})$ and cod f = dom g,
$$F(g \circ f) = F(g) \circ F(f).$$

FUNCTORS

The main action of a functor is visualized with the following commutative diagrams:





If F is a functor from $\mathbf{C}$ to $\mathbf{D}$, we write $F : \mathbf{C} \to \mathbf{D}$ or $\mathbf{C} \xrightarrow{F} \mathbf{D}$.

HASKELL EXAMPLE

Recall that $\mathcal{L}_{\text{Int}} = \langle [\text{Int}], (++), [\,] \rangle$ is a monoid, and thus a category.

Similarly, $\mathcal{L}_{\text{Char}} = \langle [\text{Char}], (++), [\,] \rangle$ is a category.

THE ASCII NUMBER FUNCTOR

Let *f* :: [Char] -> [Int] be defined as follows:

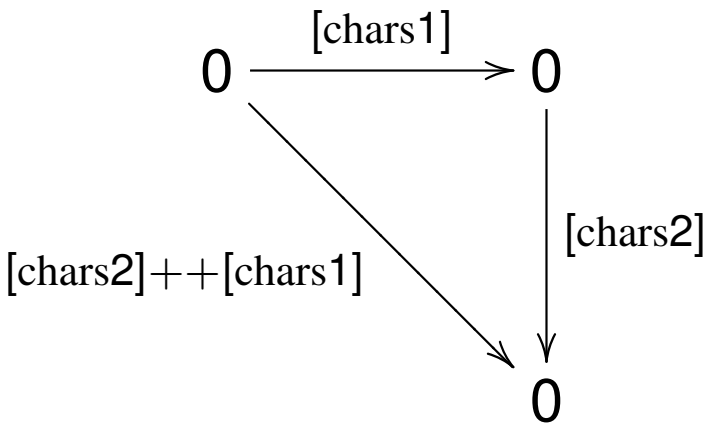
- f ([ ]) = [ ]
- f (char:chars) = [ascii(char)] ++ f([chars])

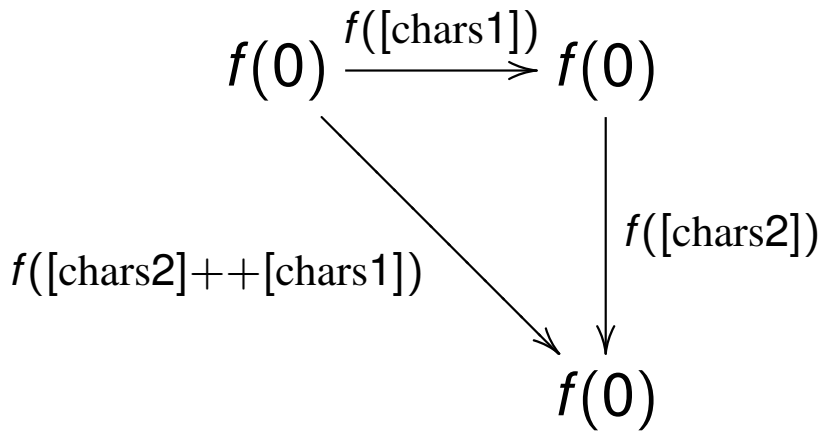
f takes a list of chars, and returns a list of ints. The *i*th element of f([chars]) is the ascii number of the *i*th char in [chars].

HASKELL EXAMPLE

We need to verify that

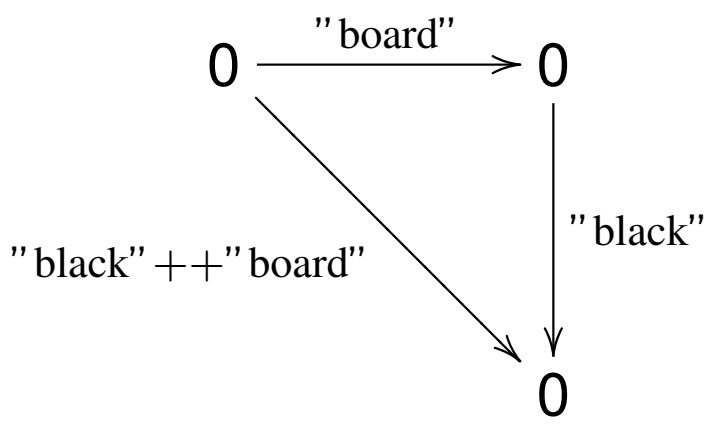
$$f\ (\ [chars2] ++ [chars1]\ ) = f\ (\ [chars2]\ ) ++ f\ (\ [chars1]\ )$$

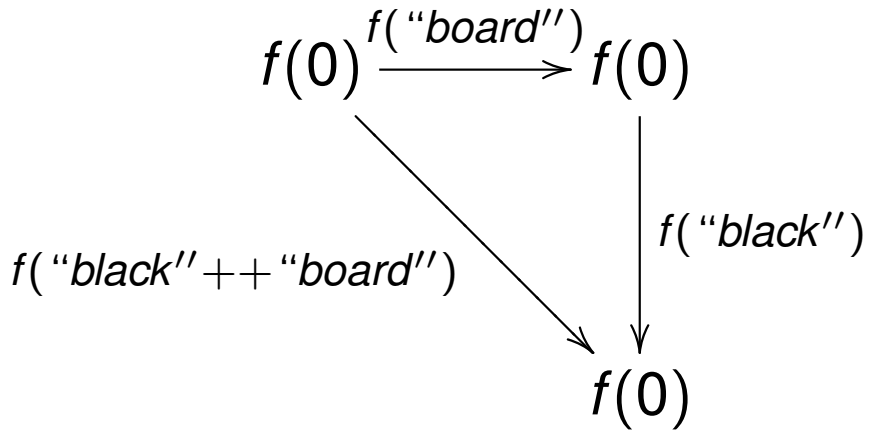




HASKELL EXAMPLE

Instance “blackboard”:





- f (“black”) = [98,108,97,99,107]
- f (“board”) = [98,111,97,114,100]
- f (“black” ++ “board”) = [98,108,97,99,107,98,111,97,114,100] = f (“blackboard”)

MONOID HOMOMORPHISMS

We can generalize the previous example.

MONOID HOMOMORPHISM

Let $\mathcal{M} = \langle M, *_M, e_M \rangle$ and $\mathcal{N} = \langle N, *_N, e_N \rangle$ be monoids. A *monoid homomorphism* is a function $\phi : \mathcal{M} \to \mathcal{N}$ such that

- $\phi(e_M) = e_N$;
- $\phi(a *_M b) = \phi(a) *_N \phi(b)$.

The monoid homomorphisms are exactly the functors between monoids (when monoids are viewed as categories).

MONOTONIC FUNCTIONS

We can treat preorders in similar fashion.

MONOTONIC FUNCTIONS

Let $\mathcal{P} = \langle P, \sqsubseteq_P \rangle$ and $\mathcal{Q} = \langle Q, \sqsubseteq_Q \rangle$ be (pre)orders. A *monotonic function* is a function $f : \mathcal{P} \to \mathcal{Q}$ such that

$$f(a \sqsubseteq_P b) = f(a) \sqsubseteq_Q f(b).$$

The monotonic functions are exactly the functors between preorders (when preorders are viewed as categories).

FUNCTORS

IDENTITY FUNCTOR

The *identity functor* on a category $\mathbf{C}$ is the functor $1_{\mathbf{C}} : \mathbf{C} \to \mathbf{C}$ such that

- for all $a \in Ob(\mathbf{C})$, $1_{\mathbf{C}}(a) = a$.
- for all $f \in Ar(\mathbf{C})$, $1_{\mathbf{C}}(f) = f$.

FUNCTORS

Since functors are functions we immediately have the following:

FUNCTOR COMPOSITION

Given functors $F : \mathbf{A} \to \mathbf{B}$ and $G : \mathbf{B} \to \mathbf{C}$, the composite of F and G, denoted $G \circ F$, is the functor defined as

- $(G \circ F)(a) = G(F(a))$, for all $a \in Ob(\mathbf{A})$;
- $(G \circ F)(f) = G(F(f))$, for all $f \in Ar(\mathbf{A})$.

For all functors $F : \mathbf{A} \to \mathbf{B}$, $G : \mathbf{B} \to \mathbf{C}$, and $H : \mathbf{C} \to \mathbf{D}$, we have $H \circ (G \circ F) = (H \circ G) \circ F$.

HASKELL EXAMPLE

$\mathcal{L}_{\text{Bool}} = \langle [\text{Bool}], (++), [\,] \rangle$ is a category.

THE IS98 FUNCTOR

Let *g* :: [Int] -> [Bool] be defined as follows:

- g ([ ]) = [ ]
- g (int:ints) = [is98(int)] ++ g([ints])

g takes a list of ints, and returns a list of bools. The *i*th element of g([ints]) is True if and only if the *i*th int in [ints] is 98.

HASKELL EXAMPLE

In Haskell syntax, the composition operator $\circ$ is denoted by .

COMPOSITION

Since g and f are functors, then so is

$$h = g.f :: [\text{Char}] \text{ -> } [\text{Bool}]$$

h takes a list of chars, and returns a list of bools. The ith element of h([chars]) is True if and only if the ith char in [chars] is 'b'.

h ("tabby") = g.f ("tabby") = [False,False,True,True,False]

CATEGORIES OF CATEGORIES

We can again generalize the previous example.

Since functor composition is associative, and identity functors satisfy the identity law for arrows, we can treat categories as objects and functors as arrows, yielding categories of categories.

CATEGORY OF MONOIDS

CATEGORY OF MONOIDS

The category of monoids **Mon** is defined as follows:

- $Ob(\mathbf{Mon}) = \{\mathcal{M} \mid \mathcal{M} \text{ is a monoid}\}$,
- $Ar(\mathbf{Mon}) = \{\phi \mid \phi \text{ is a monoid homomorphism}\}$,
- $\phi \circ \psi$ is function composition,
- $\mathrm{id}_{\mathcal{M}}$ is the identity function on M, for $\mathcal{M} \in Ob(\mathbf{Mon})$.

CATEGORY OF PREORDERS

CATEGORY OF PREORDERS

The category of preorders **PreOrd** is defined as follows:

- $Ob(\mathbf{PreOrd}) = \{\mathcal{P} \mid \mathcal{P} \text{ is a preorder}\}$,
- $Ar(\mathbf{PreOrd}) = \{f \mid f \text{ is a monotonic function on preorders}\}$,
- $f \circ g$ is function composition,
- $\mathrm{id}_{\mathcal{P}}$ is the identity function on P, for $\mathcal{P} \in Ob(\mathbf{PreOrd})$.

FUNCTOR CATEGORIES

LOOKING AHEAD

- We can define functors between categories of categories, yielding an even higher level of abstraction.
- At this level, the functors themselves are objects. We need to define the arrows.

OUTLINE

NATURAL TRANSFORMATIONS

The arrows of functor categories are called *natural transformations*.

Natural transformations are used in defining monads (next week's topic). We simply provide definitions and examples. The programming will be discussed next week.

Natural Transformations

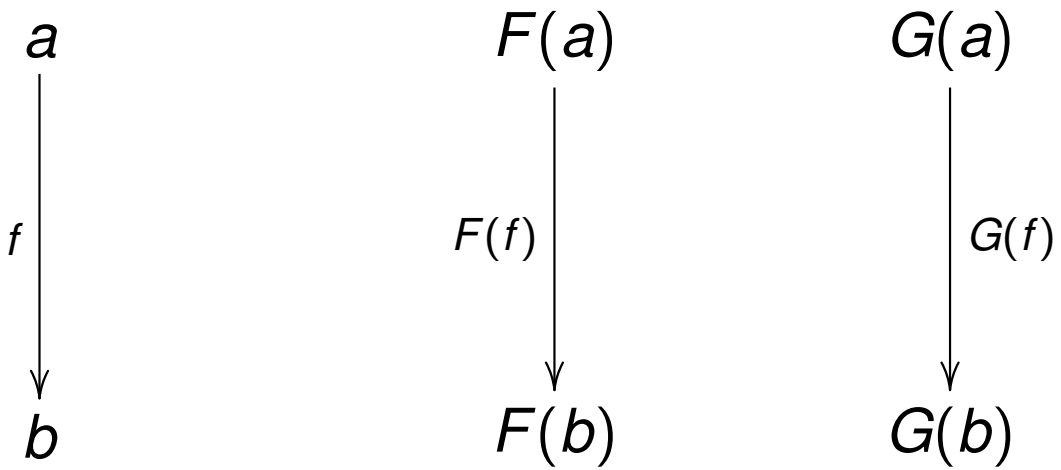
Lead in

Let **C** and **D** be categories, and let $F : \mathbf{C} \to \mathbf{D}$ and $G : \mathbf{C} \to \mathbf{D}$ be functors from **C** to **D**.

- The images of F and G are representations of the category **C** within the category **D**.
- We want to translate the representation F yields onto the representation that G yields.
- We want to do so in a way that preserves the structure of the representation that F yields.

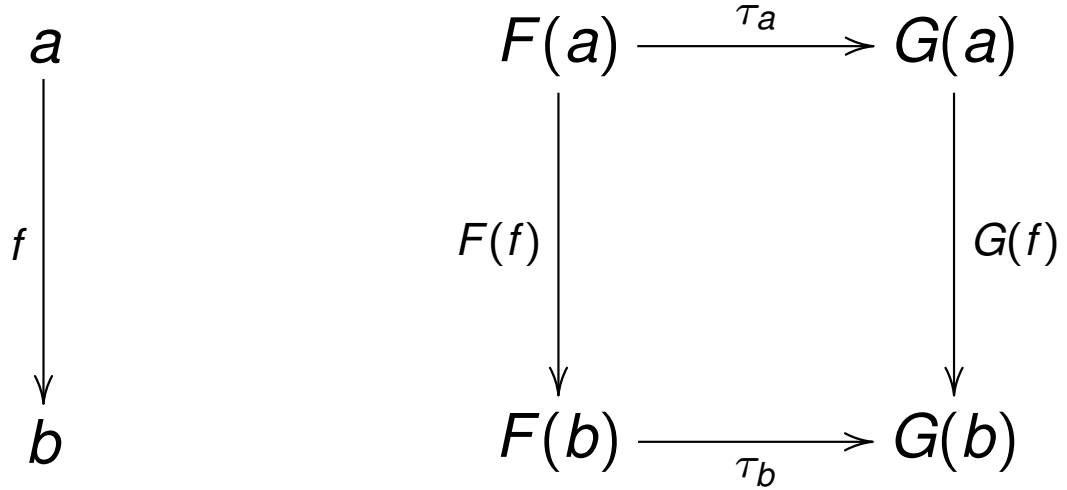
NATURAL TRANSFORMATIONS

Consider $a, b \in Ob(\mathbf{C})$ and $f \in Ar(\mathbf{C})$ where



NATURAL TRANSFORMATIONS

We translate $F(a)$ onto $G(a)$ and $F(b)$ onto $G(b)$ using arrows in **D**. Call these arrows $\tau_a : F(a) \to G(a)$ and $\tau_b : F(b) \to G(b)$.



NATURAL TRANSFORMATIONS

Let **C** and **D** be categories, and let $F : \mathbf{C} \to \mathbf{D}$ and $G : \mathbf{C} \to \mathbf{D}$ be functors from **C** to **D**.

NATURAL TRANSFORMATIONS

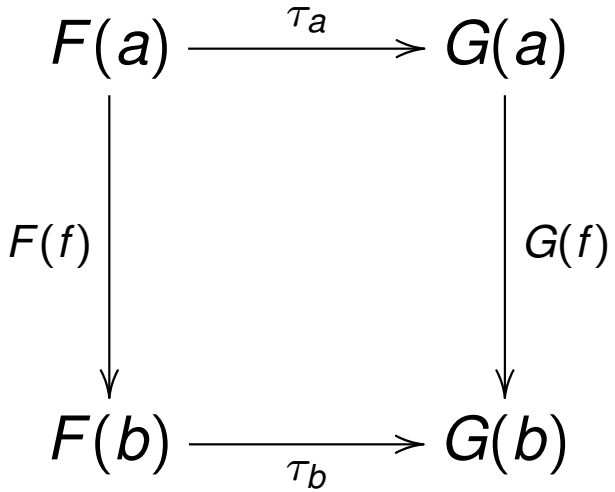
A *natural transformation* from F to G is a collection of arrows τ, contained in $Ar(\mathbf{D})$, such that:

- for each object $a \in Ob(\mathbf{C})$, there is an arrow $\tau_a : F(a) \to G(a)$ in τ, called a *component* of τ, such that
  - for any arrow $f : a \to b$ in $Ar(\mathbf{C})$, given $\tau_b : F(b) \to G(b) \in \tau$, we have

$$G(f) \circ \tau_a = \tau_b \circ F(f).$$

NATURAL TRANSFORMATIONS

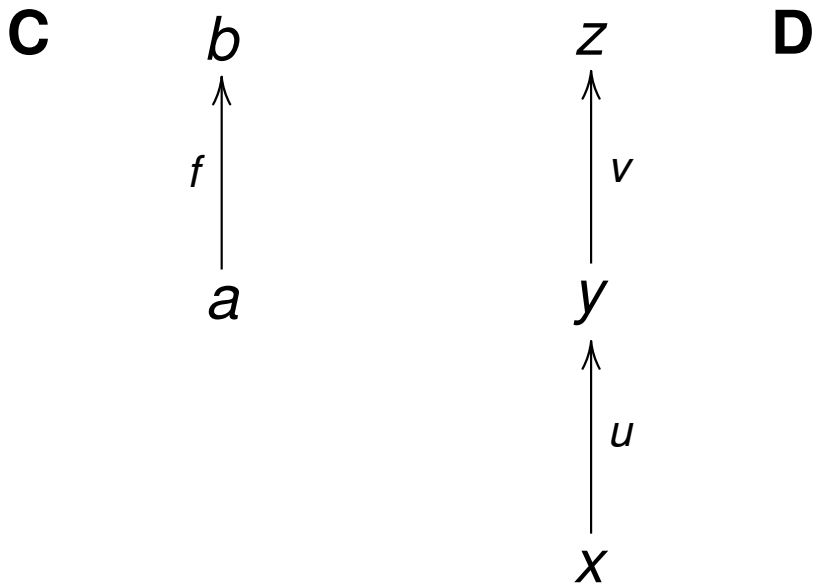
That is, the following diagram commutes:



If τ is a natural transformation from F to G, we write $\tau : F \to G$, or $F \xrightarrow{\tau} G$,

NATURAL TRANSFORMATIONS

Let **C** and **D** be the categories represented by the following diagrams (identities and compositions are suppressed):



NATURAL TRANSFORMATIONS

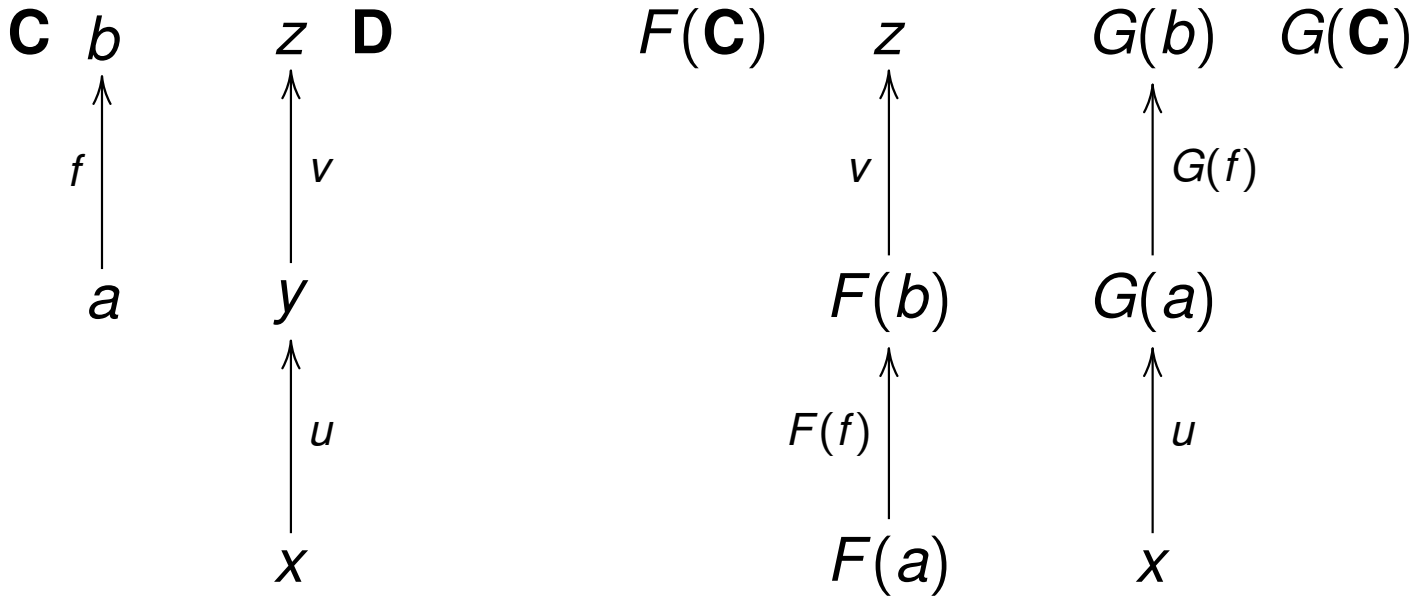
Let $F : \mathbf{C} \to \mathbf{D}$ and $G : \mathbf{C} \to \mathbf{D}$ be functors from $\mathbf{C}$ to $\mathbf{D}$, defined as follows:

$$\begin{array}{ll} F(a) = x & G(a) = y \\ F(b) = y & G(b) = z \\ F(f) = u & G(f) = v \end{array}$$

The reader may infer the remaining arrow assignments.

NATURAL TRANSFORMATIONS

That is, we have the following representations of **C** in **D**:



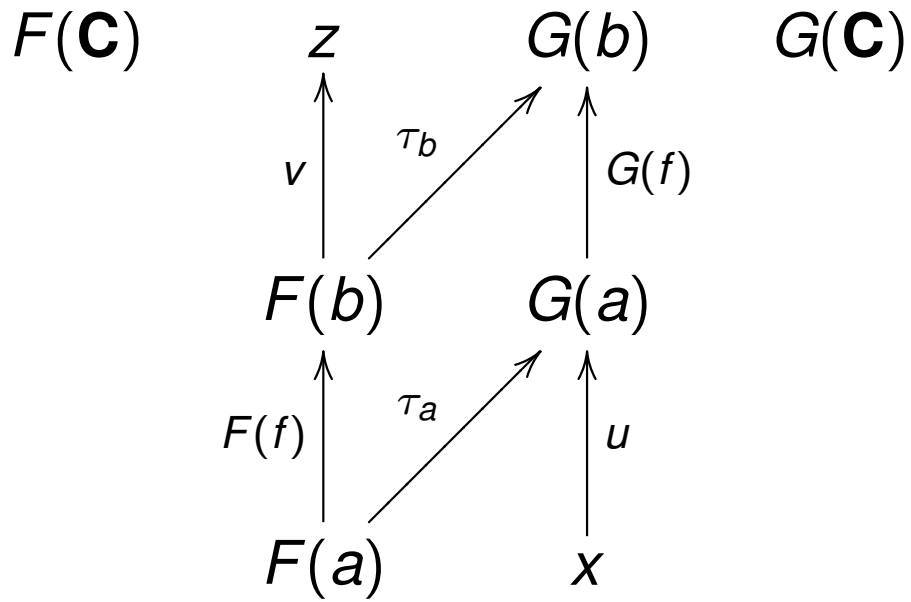
NATURAL TRANSFORMATIONS

Now, we define a natural transformation τ from F to G as the collection of arrows:

$$\tau_a = u$$
$$\tau_b = v.$$

NATURAL TRANSFORMATIONS

That is, τ is the following "shift":



NATURAL TRANSFORMATIONS

IDENTITY TRANSFORMATION

The *identity transformation* on a functor $F : \mathbf{C} \to \mathbf{D}$ is the natural transformation $1_F : F \to F$ such that for all $a \in Ob(\mathbf{C})$,

$$\tau_a = \mathrm{id}_{F(a)} : F(a) \to F(a).$$

NATURAL TRANSFORMATIONS

NATURAL TRANSFORMATION COMPOSITION

Let F, G, and H be functors from category **C** to category **D**, and let $\tau : F \to G$ and $\sigma : G \to H$ be natural transformations. The *composite* of τ and σ, denoted $\sigma \circ \tau$, is the collection of arrows $(\sigma \circ \tau)_a = \sigma_a \circ \tau_a$ for all $a \in Ob(\mathbf{C})$.

- Natural transformation composition is, of course, associative. The proof is left to the enthusiastic reader.
- Composition together with identity gives us the functor category $\mathbf{D}^{\mathbf{C}}$ – the category of all functors from **C** to **D**.

For Great Good

The Haskell

More information on Haskell Syntax and Theory:

- The Haskell: http://www.haskell.org
- Haskell Beginners: http://www.haskell.org/mailman/listinfo/beginners
- Haskell Cafe: http://www.haskell.org/mailman/listinfo/haskell-cafe