Category theory and its applications - ITI9200

Natural Transformations

February 15, 2021

Introducing Natural Transformations

Big picture

Natural transformations are morphisms of functors.

"category" has been defined in order to be able to define "functor" and "functor" has been defined in order to be able to define "natural transformation"[1]

Saunders Mac Lane and Samuel Eilenberg[2]

[1] MacLane, S. (1965). Categorical algebra. Bulletin of the American Mathematical Society, 71(1), 40-106.

[2] Eilenberg, S., & MacLane, S. (1945). General theory of natural equivalences. Transactions of the American Mathematical Society, 58(2), 231-294.

Natural transformations I: Parametric polymorphism

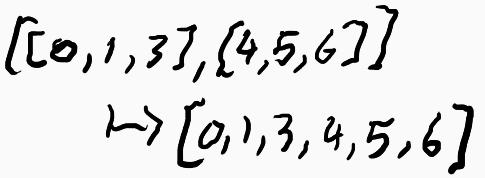
In the previous lecture we saw the functor `List : Type -> Type`.

Natural transformations I: Parametric polymorphism

In the previous lecture we saw the functor `List : Type -> Type`.

We can also write *functions parametric* in a type variable `a`, for example:

```
flatten : List (List a) -> List a
flatten [] = []
flatten (x :: xs) = x ++ flatten xs
```



We can think of such a function as a family of functions, one for each type `a`.

Natural transformations I: Parametric polymorphism

In the previous lecture we saw the functor `List : Type -> Type`.

We can also write *functions parametric* in a type variable `a`, for example:

```
flatten : List (List a) -> List a
flatten [] = []
flatten (x :: xs) = x ++ flatten xs
```

We can think of such a function as a family of functions, one for each type `a`.

Moreover, these functions behave *uniformly* with respect to the type of elements: the behaviour does not (cannot) *depend* on the type `a`.

Natural transformations I: Parametric polymorphism

In the previous lecture we saw the functor `List : Type -> Type`.

We can also write *functions parametric* in a type variable `a`, for example:

```
flatten : List (List a) -> List a
flatten [] = []
flatten (x :: xs) = x ++ flatten xs
```

We can think of such a function as a family of functions, one for each type `a`.

Moreover, these functions behave *uniformly* with respect to the type of elements: the behaviour does not (cannot) *depend* on the type `a`.

In other words, whenever we have some `f : a -> b`, it makes no difference if we apply `f` then `flatten`, or `flatten` then apply `f`.

Naturality condition

```
flatten : List (List a) -> List a
flatten [] = []
flatten (x :: xs) = x ++ flatten xs

fmap f . flatten = flatten . fmap f
```

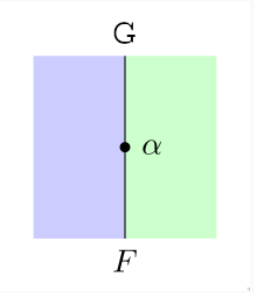
This condition on the uniformity of behaviour, parametricity, is an instance of the general notion of *naturality*.

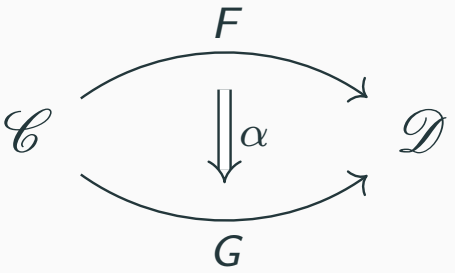
$$\begin{array}{ccc} [[a_0^0, \ldots, a_{n_0}^0], \ldots, [a_0^m, \ldots, a_{n_m}^m]] & \xmapsto{\text{flatten}_a} & [a_0^0, \ldots, a_{n_0}^0, \ldots, a_0^m, \ldots, a_{n_m}^m] \\ \downarrow \text{List} \circ \text{List} f & & \downarrow \text{List} f \\ [[b_0^0, \ldots, b_{n_0}^0], \ldots, [b_0^m, \ldots, b_{n_m}^m]] & \xmapsto{\text{flatten}_b} & [b_0^0, \ldots, b_{n_0}^0, \ldots, b_0^m, \ldots, b_{n_m}^m] \end{array}$$

$$\text{flatten} : \text{List} \circ \text{List} \Rightarrow \text{List}$$

Natural Transformations: Formal definition

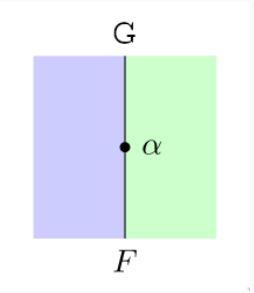
Definition
Given two functors $F, G : \mathscr{C} \rightrightarrows \mathscr{D}$ (jargon: *parallel* functors), a natural transformation $\alpha : F \Rightarrow G$ is defined by:

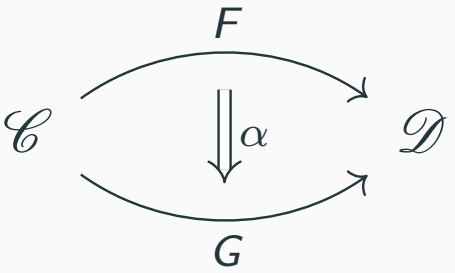




Natural Transformations: Formal definition

Definition
Given two functors $F, G : \mathscr{C} \rightrightarrows \mathscr{D}$ (jargon: *parallel* functors), a natural transformation $\alpha : F \Rightarrow G$ is defined by:

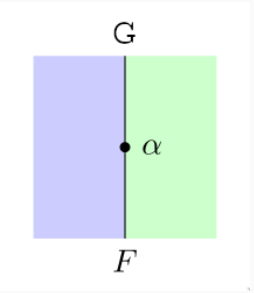


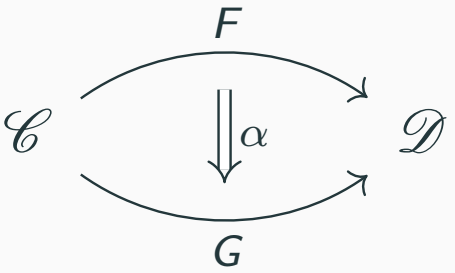


- A morphism $\alpha_c : Fc \to Gc$ in $\mathscr{D}$, for every object $c : \mathscr{C}$, called the *components* of α

Natural Transformations: Formal definition

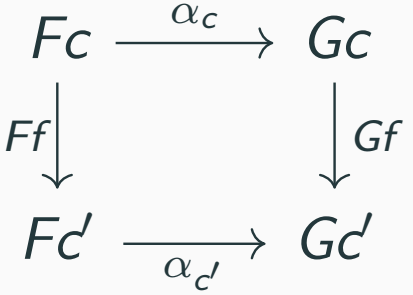
Definition
Given two functors $F, G : \mathscr{C} \rightrightarrows \mathscr{D}$ (jargon: *parallel* functors), a natural transformation $\alpha : F \Rightarrow G$ is defined by:





- A morphism $\alpha_c : Fc \to Gc$ in $\mathscr{D}$, for every object $c : \mathscr{C}$, called the *components* of α
- such that for every morphism $f : c \to c'$ in $\mathscr{C}$, the following square commutes (*naturality condition*):

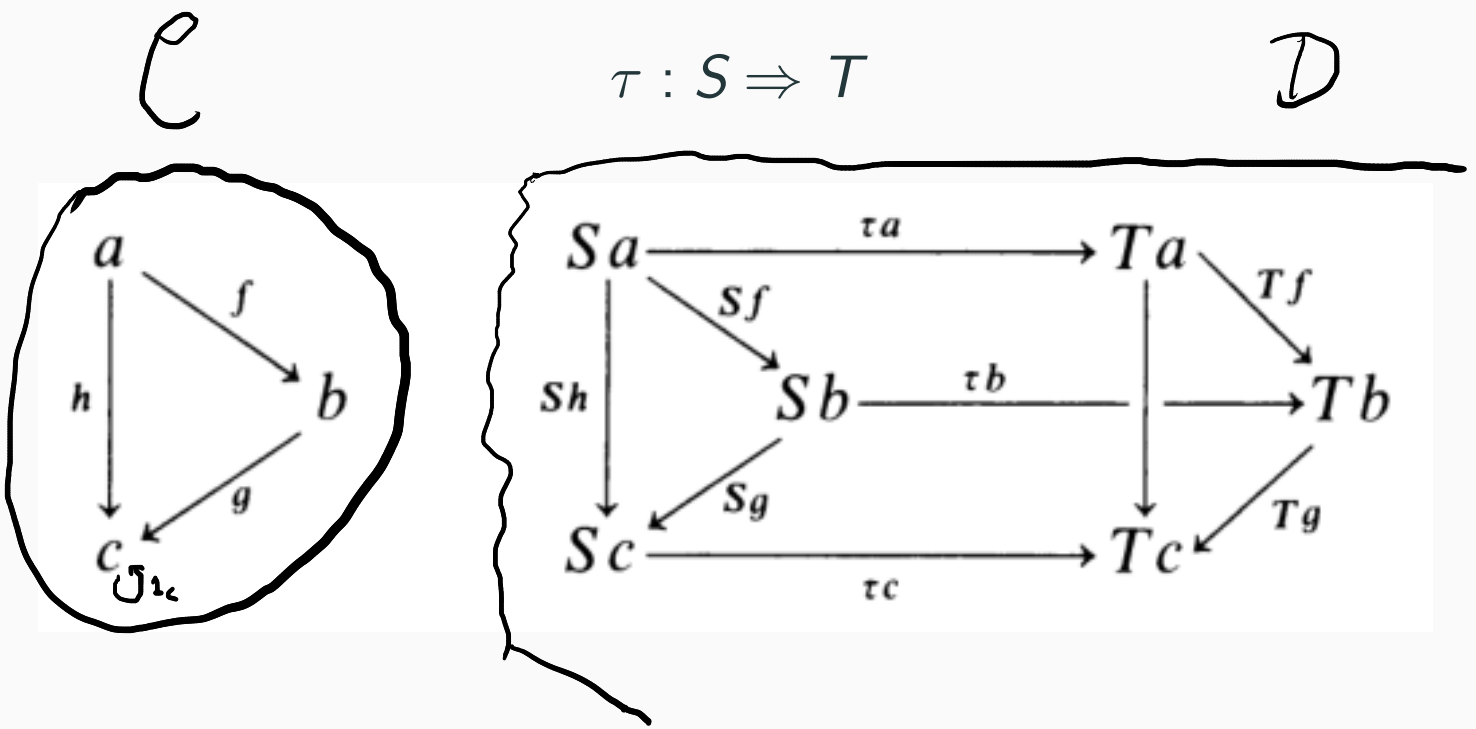
$Gf \circ \alpha_c = \alpha_{c'} \circ Ff$



Intution 1: morphing images of functors

$$S, T : \mathscr{C} \rightrightarrows \mathscr{D}$$

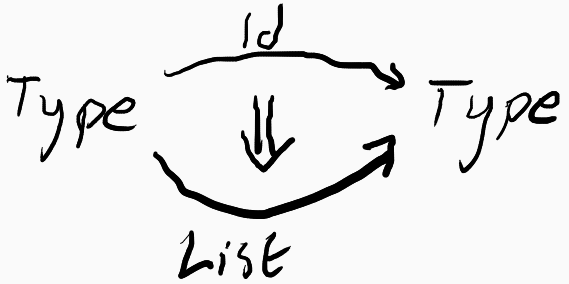
$$\tau : S \Rightarrow T$$

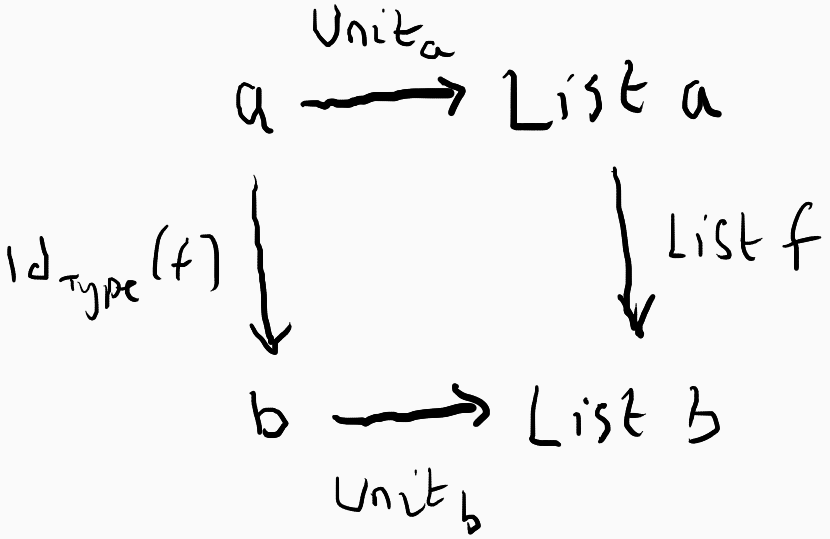


More examples of natural transformations

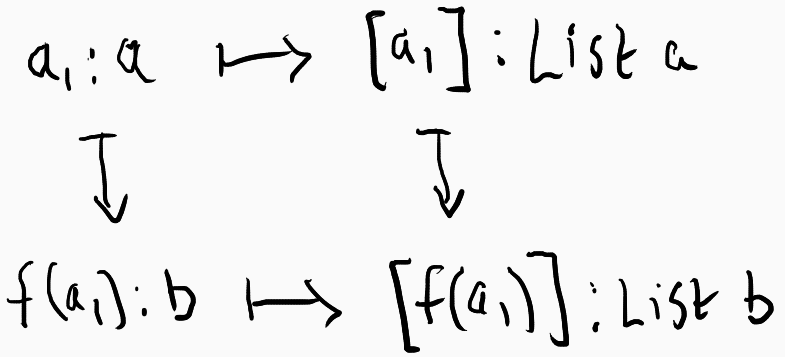
```
unit : a -> List a
unit x = [x]
```

$\text{unit} : \text{Id}_{\text{Type}} \Rightarrow \text{List}$





$f : a \to b$

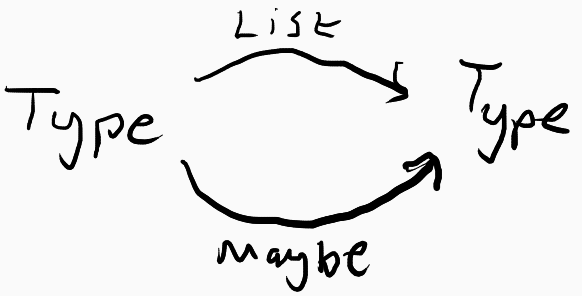


More examples of natural transformations

```
head : List a -> Maybe a
head [] = Nothing
head (x :: xs) = Just x
```

$$\begin{array}{ccc} \mathrm{List}\ a & \xrightarrow{\mathrm{head}_a} & \mathrm{Maybe}\ a \\ \downarrow{\scriptstyle \mathrm{List}\ f} & & \downarrow{\scriptstyle \mathrm{Maybe}\ f} \\ \mathrm{List}\ b & \xrightarrow[\mathrm{head}_b]{} & \mathrm{Maybe}\ b \end{array}$$

$\mathrm{head} : \mathrm{List} \Rightarrow \mathrm{Maybe}$



$\mathrm{Maybe} : a \mapsto a + \{*\}$

$f : a \to b \mapsto f' * \mapsto *$

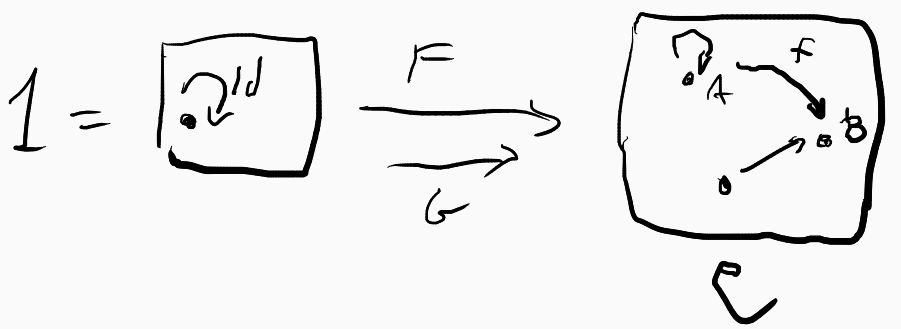
$f' a \mapsto f a$

$$\begin{array}{ccc} [\,] & \longmapsto & * \\ \downarrow & & \downarrow \\ [\,] & \longmapsto & * \end{array} \qquad \begin{array}{ccc} [a_1, \ldots, a_m] & \longmapsto & a_1 \\ \downarrow & & \downarrow \\ [b_1, \ldots, b_n] & \longmapsto & b_1 \end{array}$$

$f(a_i)$

More examples of natural transformations

For $F, G : 1 \rightrightarrows \mathscr{C}$, a natural transformation $\alpha : F \Rightarrow G$ is just a morphism in $\mathscr{C}$.



"$F(id) = Id$"

$F(\cdot) = A \qquad G(\cdot) = B$

$F(id_\cdot) = Id_A \qquad \ldots$

$\alpha : F \Rightarrow G$

$\alpha_\cdot : F(\cdot) \to G(\cdot) \qquad \alpha_\cdot := f$

$$\begin{array}{ccc} F_\cdot & \xrightarrow{\alpha_\cdot} & G_\cdot \\ {\scriptstyle F(id_\cdot)}\downarrow & & \downarrow{\scriptstyle G(id)} \\ F_\cdot & \xrightarrow[\alpha_\cdot]{} & G_\cdot \end{array} \qquad \begin{array}{ccc} A & \xrightarrow{f} & B \\ {\scriptstyle id_A}\downarrow & & \downarrow{\scriptstyle id_B} \\ A & \xrightarrow[f]{} & B \end{array}$$

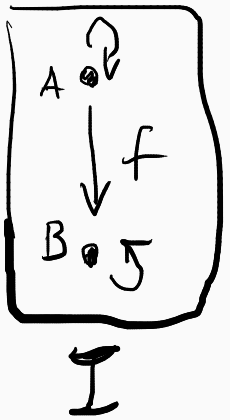
$id_B \circ f = f$

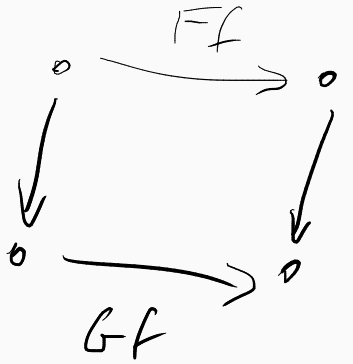
$id_B \circ f = f \circ id_A$

$f = f$

More examples of natural transformations

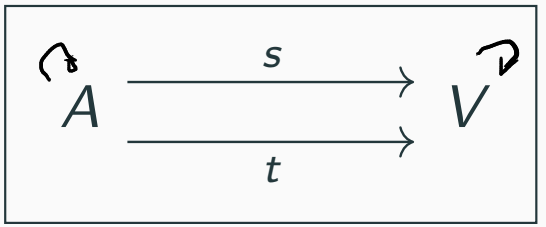
For $F, G : I \rightrightarrows \mathscr{C}$, a natural transformation $\alpha : F \Rightarrow G$ is just a commutative square in $\mathscr{C}$.





Homomorphisms as natural transformations of models

Consider the following category (identity morphisms are not drawn):

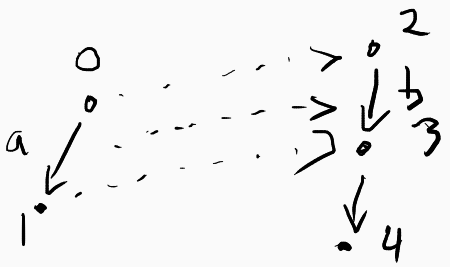
$\mathscr{C} =$ 

$s \neq t$

Then a functor $\mathcal{G} : \mathscr{C} \to \mathrm{Set}$ is a graph: a set $\mathcal{G}(A)$ of "arrows", a set $\mathcal{G}(V)$ of "vertices", and two functions $\mathcal{G}(s), \mathcal{G}(t) : \mathcal{G}(A) \to \mathcal{G}(V)$ that tell us the source and target of each arrow.

Sometimes we call $\mathscr{C}$ the *theory* (of graphs), and functors $\mathscr{C} \to \mathrm{Set}$ *models* (of the theory of graphs).

Homomorphisms as natural transformations of models



$$\mathscr{C} = \boxed{A \underset{t}{\overset{s}{\rightrightarrows}} V}$$

Consider two functors (graphs) $\mathcal{G}, \mathcal{G}' : \mathscr{C} \to \mathrm{Set}$. A natural transformation $\alpha : \mathcal{G} \Rightarrow \mathcal{G}'$ is a homomorphism of graphs. Naturality says that the image of the source of each arrow must be the source of the image of each arrow (and likewise for targets):

$$\begin{array}{ccc} \mathcal{G}A & \xrightarrow{\alpha_A} & \mathcal{G}'A \\ {\scriptstyle Gs}\downarrow & & \downarrow{\scriptstyle G's} \\ \mathcal{G}V & \xrightarrow[\alpha_V]{} & \mathcal{G}'V \end{array} \qquad \begin{array}{ccc} a & \mapsto & b \\ \mathcal{G}A & \xrightarrow{\alpha_A} & \mathcal{G}'A \\ {\scriptstyle Gt}\downarrow & & \downarrow{\scriptstyle G't} \\ 1 \quad \mathcal{G}V & \xrightarrow[\alpha_V]{} & \mathcal{G}'V \quad 3 \end{array}$$

This generalizes to all the familiar algebraic structures: monoids, groups, etc., which also have corresponding "theories": homomorphisms are natural transformation of their models.

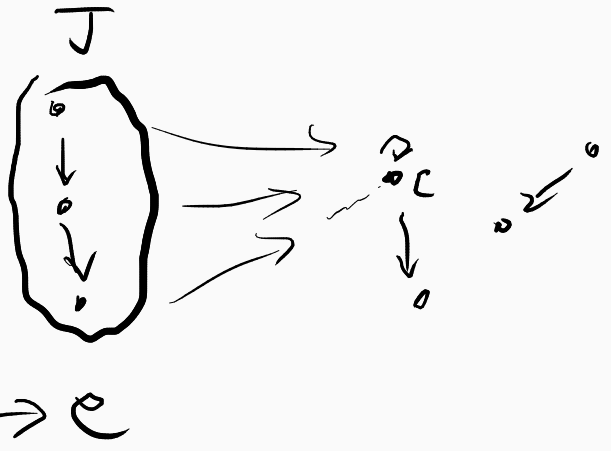
Examples of Natural Transformations: Cones

An important example of functors are the *constant functors*.

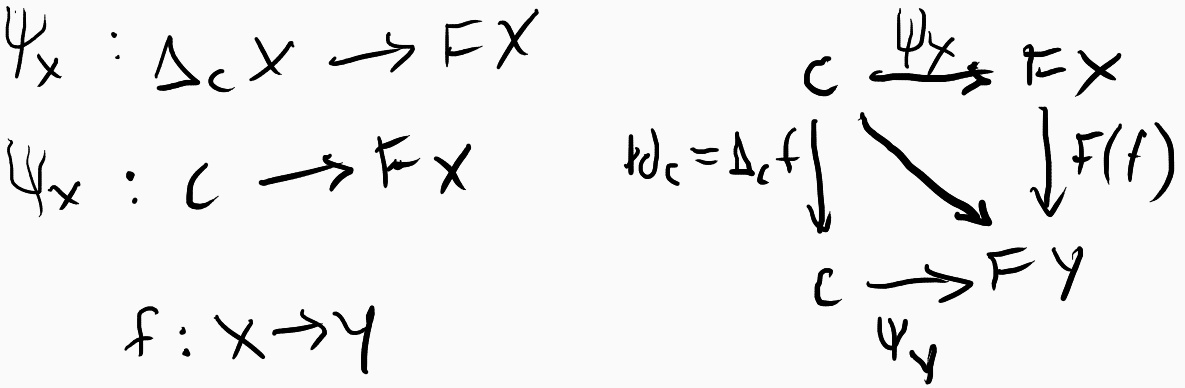
For categories $\mathscr{J}$ and $\mathscr{C}$, and any object $c : \mathscr{C}$, we can define the *constant functor at c*, $\Delta_c : \mathscr{J} \to \mathscr{C}$ by:

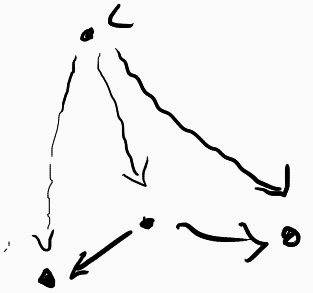
$$\Delta_c(X) \mapsto c$$

$$\Delta_c(f) \mapsto 1_c$$



Definition
A natural transformation $\psi : \Delta_c \Rightarrow F$ is called a *cone over F*.





Natural isomorphism (aka natural equivalence)

Let $\alpha : F \Rightarrow G$ be a natural transformation. If $\alpha_X : FX \xrightarrow{\sim} GX$ is an isomorphism for every X, then the inverses $\alpha_X^{-1} : GX \to FX$ define a natural transformation $\alpha^{-1} : G \Rightarrow F$.

$\alpha : G \cong F$

We call such α *natural isomorphisms* (or *natural equivalences*).

"$f f^{-1} = 1$
$f^{-1} f = 1$"

$GX \cong FX$
"naturally in X"

$$Gf \circ \alpha_X = \alpha_Y \circ Ff \tag{1}$$

$$Gf \circ \alpha_X \circ \alpha_X^{-1} = \alpha_Y \circ Ff \circ \alpha_X^{-1} \tag{2}$$

$$Gf \circ 1_{GX} = \alpha_Y \circ Ff \circ \alpha_X^{-1} \tag{3}$$

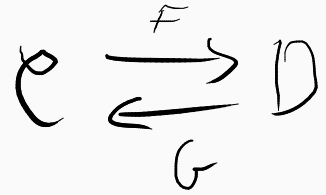
$$\alpha_Y^{-1} \circ Gf = \alpha_Y^{-1} \circ \alpha_Y \circ Ff \circ \alpha_X^{-1} \tag{4}$$

$$\alpha_Y^{-1} \circ Gf = 1_{FY} \circ Ff \circ \alpha_X^{-1} \tag{5}$$

$$\alpha_Y^{-1} \circ Gf = Ff \circ \alpha_X^{-1} \tag{6}$$

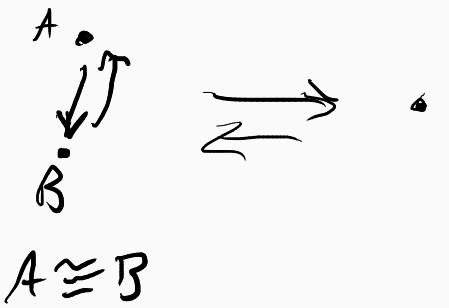
Equivalence of categories: motivation

Recall that given a monoid $(M, \cdot)$ we can form a category $C(M)$ with one object, say M, and whose morphisms $M \to M$ are elements of M and with composition given by $\cdot$.



$FG \cong 1_D$

$GF \cong 1_C$



$A \cong B$

$FG \Rightarrow 1_D$

Equivalence of categories: motivation

Recall that given a monoid $(M, \cdot)$ we can form a category $C(M)$ with one object, say M, and whose morphisms $M \to M$ are elements of M and with composition given by $\cdot$.

Recall that a homomorphism of monoids $(M, \cdot) \to (N, \star)$ is a function $h : M \to N$, such that $h(x + y) = h(x) + h(y)$ and $h(1_M) = 1_N$.

Equivalence of categories: motivation

Recall that given a monoid $(M, \cdot)$ we can form a category $C(M)$ with one object, say M, and whose morphisms $M \to M$ are elements of M and with composition given by $\cdot$.

Recall that a homomorphism of monoids $(M, \cdot) \to (N, \star)$ is a function $h : M \to N$, such that $h(x + y) = h(x) + h(y)$ and $h(1_M) = 1_N$.

Then for every homomorphism of monoids h, we have a functor $H : C(M) \to C(N)$ that maps $\mathcal{M}$ to $\mathcal{N}$, and a morphism $m : \mathcal{M} \to \mathcal{M}$ to the arrow $h(m) : \mathcal{N} \to \mathcal{N}$.

Equivalence of categories: motivation

Recall that given a monoid $(M, \cdot)$ we can form a category $C(M)$ with one object, say M, and whose morphisms $M \to M$ are elements of M and with composition given by $\cdot$.

Recall that a homomorphism of monoids $(M, \cdot) \to (N, \star)$ is a function $h : M \to N$, such that $h(x + y) = h(x) + h(y)$ and $h(1_M) = 1_N$.

Then for every homomorphism of monoids h, we have a functor $H : C(M) \to C(N)$ that maps $\mathcal{M}$ to $\mathcal{N}$, and a morphism $m : \mathcal{M} \to \mathcal{M}$ to the arrow $h(m) : \mathcal{N} \to \mathcal{N}$.

Now C itself defines a functor, $C : \mathrm{Mon} \to \mathrm{Ooc}$ (check identities and composition).

Equivalence of categories: motivation

Recall that given a monoid $(M, \cdot)$ we can form a category $C(M)$ with one object, say M, and whose morphisms $M \to M$ are elements of M and with composition given by $\cdot$.

Recall that a homomorphism of monoids $(M, \cdot) \to (N, \star)$ is a function $h : M \to N$, such that $h(x + y) = h(x) + h(y)$ and $h(1_M) = 1_N$.

Then for every homomorphism of monoids h, we have a functor $H : C(M) \to C(N)$ that maps $\mathcal{M}$ to $\mathcal{N}$, and a morphism $m : \mathcal{M} \to \mathcal{M}$ to the arrow $h(m) : \mathcal{N} \to \mathcal{N}$.

Now C itself defines a functor, $C : \mathrm{Mon} \to \mathrm{Ooc}$ (check identities and composition).

We can similarly define a functor $U : \mathrm{Ooc} \to \mathrm{Mon}$ that sends $\mathscr{C}$ to the monoid $(\mathrm{Mor}(\mathscr{C}), \circ)$

Equivalence of categories: motivation

Recall that given a monoid $(M, \cdot)$ we can form a category $C(M)$ with one object, say M, and whose morphisms $M \to M$ are elements of M and with composition given by $\cdot$.

Recall that a homomorphism of monoids $(M, \cdot) \to (N, \star)$ is a function $h : M \to N$, such that $h(x + y) = h(x) + h(y)$ and $h(1_M) = 1_N$.

Then for every homomorphism of monoids h, we have a functor $H : C(M) \to C(N)$ that maps $\mathcal{M}$ to $\mathcal{N}$, and a morphism $m : \mathcal{M} \to \mathcal{M}$ to the arrow $h(m) : \mathcal{N} \to \mathcal{N}$.

Now C itself defines a functor, $C : \mathrm{Mon} \to \mathrm{Ooc}$ (check identities and composition).

We can similarly define a functor $U : \mathrm{Ooc} \to \mathrm{Mon}$ that sends $\mathscr{C}$ to the monoid $(\mathrm{Mor}(\mathscr{C}), \circ)$

Question: what is $C(U(\mathscr{C})) : \mathrm{Ooc}$? Is it just $\mathscr{C}$ again?

Equivalence of categories: motivation

Recall that given a monoid $(M, \cdot)$ we can form a category $C(M)$ with one object, say M, and whose morphisms $M \to M$ are elements of M and with composition given by $\cdot$.

Recall that a homomorphism of monoids $(M, \cdot) \to (N, \star)$ is a function $h : M \to N$, such that $h(x + y) = h(x) + h(y)$ and $h(1_M) = 1_N$.

Then for every homomorphism of monoids h, we have a functor $H : C(M) \to C(N)$ that maps $\mathcal{M}$ to $\mathcal{N}$, and a morphism $m : \mathcal{M} \to \mathcal{M}$ to the arrow $h(m) : \mathcal{N} \to \mathcal{N}$.

Now C itself defines a functor, $C : \mathrm{Mon} \to \mathrm{Ooc}$ (check identities and composition).

We can similarly define a functor $U : \mathrm{Ooc} \to \mathrm{Mon}$ that sends $\mathscr{C}$ to the monoid $(\mathrm{Mor}(\mathscr{C}), \circ)$

Question: what is $C(U(\mathscr{C})) : \mathrm{Ooc}$? Is it just $\mathscr{C}$ again?

Equivalence of categories

Definition
An equivalence of categories $\mathscr{C} \simeq \mathscr{D}$ is given by functors $F : C \rightleftarrows D : G$, and natural isomorphisms $\eta : 1_{\mathscr{C}} \cong GF$, $\epsilon : FG \cong 1_{\mathscr{D}}$.

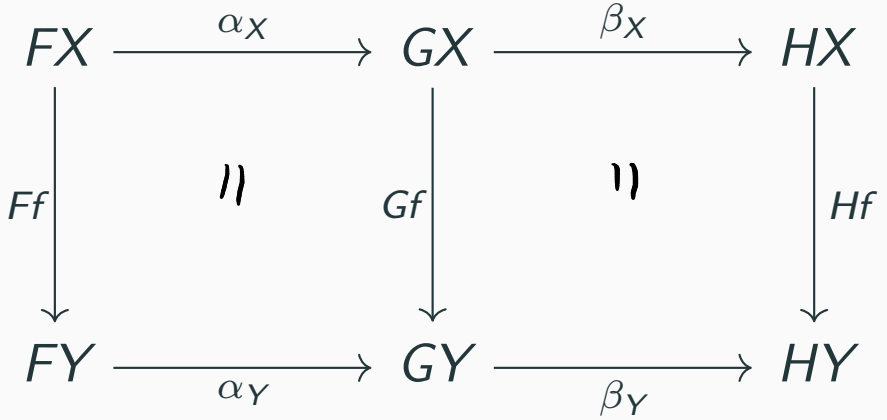
Slogan: all categorical notions should be invariant under equivalence.

Important in mathematics! Equivalences of the form $\mathscr{C} \simeq \mathscr{D}^{\mathrm{op}}$ can encapsulate profound "dualities". For example, FinSet $\simeq$ FinBool$^{\mathrm{op}}$.

Composing natural transformations I: Vertical composition

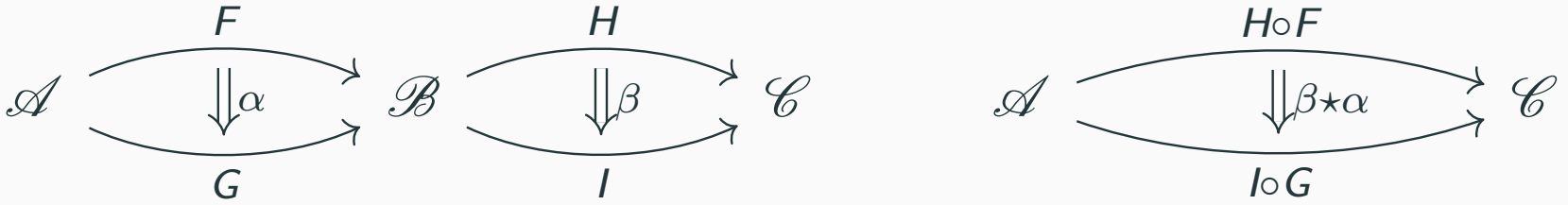
Consider functors $F, G, H : \mathscr{A} \to \mathscr{B}$, and natural transformations $\alpha : F \Rightarrow G, \beta : G \Rightarrow H$ (as on the left below), then we can form a new natural transformation $\beta \circ \alpha : F \Rightarrow H$ (as on the right):

With component at $X : \mathscr{A}$ defined by $(\beta \circ \alpha)_X : FX \to HX := \beta_X \circ \alpha_X$. That these components satisfy naturality is witnessed by the following diagram:

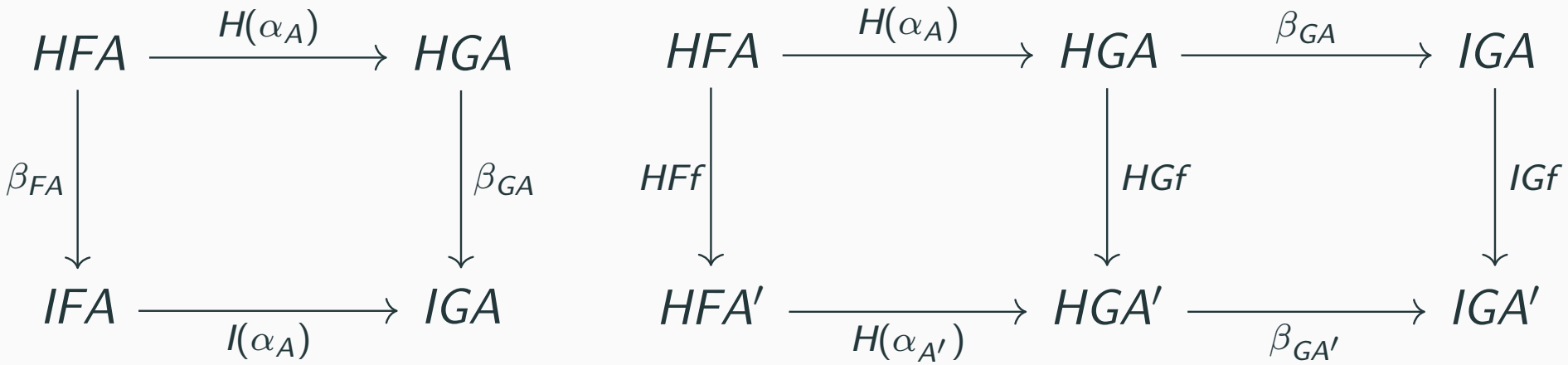


Composing natural transformations II: Horizontal composition

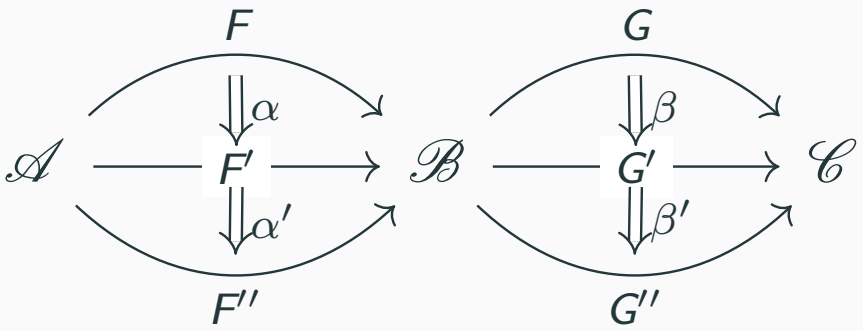
We can also compose natural transformations "horizontally". So, given the four functors and two natural transformations on the left below, we can form their "horizontal composite" on the right:



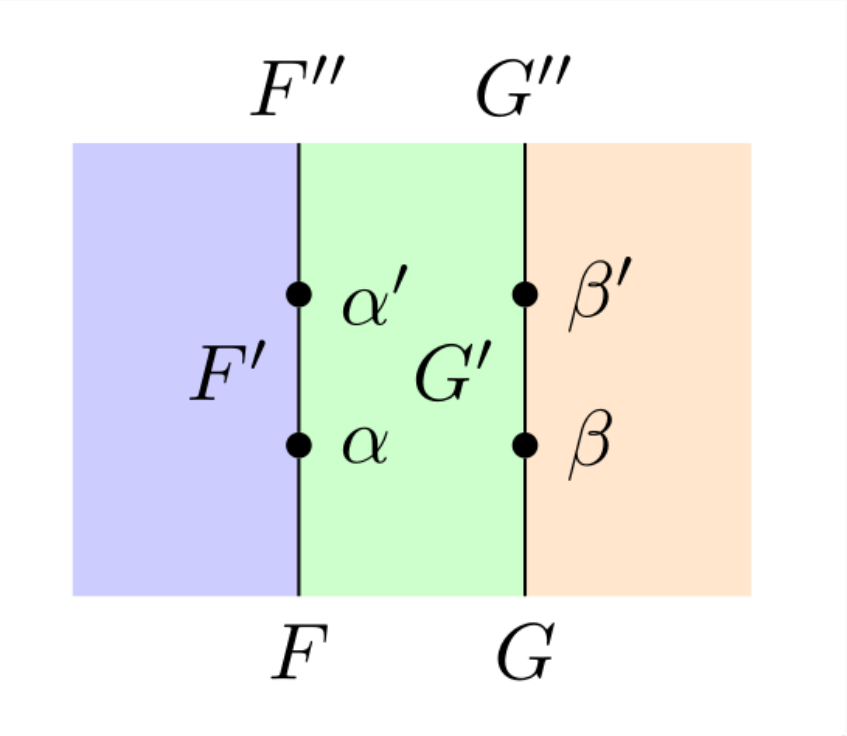
With component at $X : \mathscr{A}$ defined by
$(\beta \star \alpha)_X = \beta_{GX} \circ H(\alpha_X) = I(\alpha_X) \circ \beta_{FX}$.

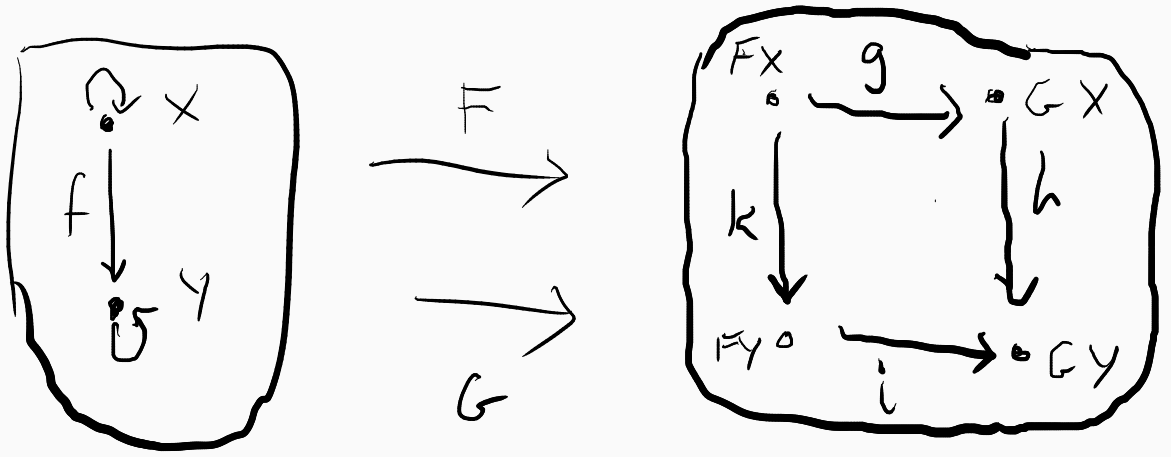


Interchange law



$$(\beta' \circ \beta) \star (\alpha' \circ \alpha) = (\beta' \star \alpha') \circ (\beta \star \alpha)$$





$h \circ g \neq i \circ k$

$Ff = k$

$Gf = h$

$\mathcal{C}$ $\mathcal{D}$

$\alpha : F \Rightarrow G$

$\alpha_X := g : FX \to GX$

$\alpha_Y := i : FY \to GY$

$Gf \circ \alpha_X \neq \alpha_Y \circ Ff$

$h \circ g \neq i \circ k$