Intro to Category Theory: Natural Transformations

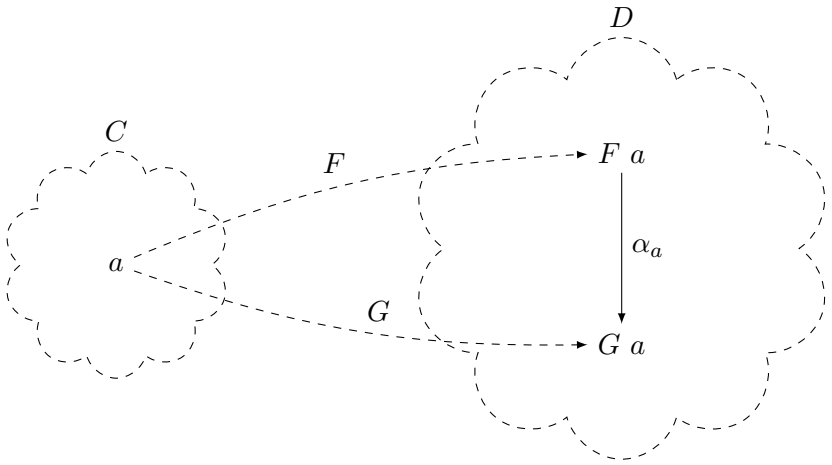
1 Natural Transformations

With category theory we first define categories, which are a way to formalize structure preserving transformations called morphisms. Then we define functors which are structure preserving transformations of categories. Now we are going to define natural transformations, which are going to be transformations of functors. It may seem that we are never going to stop. It's all about transformations on transformations forever. This is partly true. There is a branch of category theory called higher category theory that does take this perspective. However, we are not going to explore this field. Fortunately, in regular category theory the transformations stop here. That is, we are going to stop at natural transformations.

It is often remarked that categories are only defined to define functors and functors are only defined to define natural transformations. Saunders Mac Lane supposedly said "I didn't invent categories to study functors; I invented them to study natural transformations."

We shall build up the definition of a natural transformation step by step. A functor can be thought of an embedding of one category into another. Suppose we had two such embeddings $F : C \to D$ and $G : C \to D$. We would like to compare the result of these embeddings, by defining a transformation between F and G. We usually use lower case Greek letters to denote natural transformations $(\alpha, \beta, \gamma, ...)$. F and G map objects to objects and morphisms to morphisms. To define a transformation between F and G we want to have the objects and morphisms mapped by F to be related to the objects and morphisms mapped by G.

Let's start with objects. Consider an object a of C. $F\ a$ and $G\ a$ are objects of D. We can impose a traditional mapping (a function) between $F\ a$ and $G\ a$ for each a, but we didn't come all this way to revert back to set theory. We might already have a relationship between $F\ a$ and $G\ a$ as a morphism in D. Let $\alpha : F \to G$ be a natural transformation. This natural transformation picks out these morphisms in D between $F\ a$ and $G\ a$ for each a. This morphism is denoted as $\alpha_a : F\ a \to G\ a$ and is called the component of α at a.



This is what α does for objects any object, but what about morphisms? Suppose we have a morphism $f : a \to b$ of C. Because F and G are functors there necessarily are morphisms $F\ f : F\ a \to F\ b$ and $G\ f : G\ a \to G\ b$ in D. Also, if the components of a natural transformation have been defined we have morphisms $\alpha_a : F\ a \to G\ a$ and $\alpha_b : F\ b \to G\ b$ of D. Because D is a category we have two ways to get a morphism from $F\ a$ to $G\ b$. We can use α_a then $G\ f$,

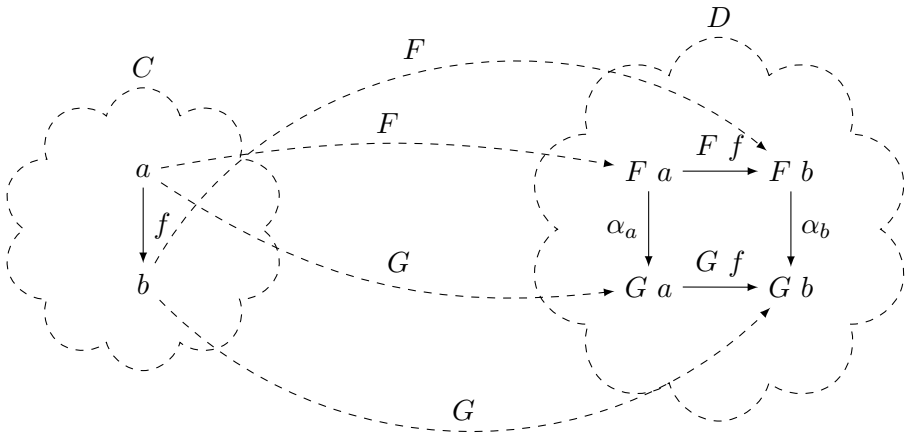
or we can take $F\ f$ and then α_b. So far, these resulting morphisms don't have to be related. $(G\ f) \circ \alpha_a : F\ a \to G\ b$ can be totally different from $\alpha_b \circ (F\ f) : F\ a \to G\ b$. However, we can relate $F\ f$ to $G\ f$ with α if we require $(G\ f) \circ \alpha_a = \alpha_b \circ (F\ f)$. This condition is known as the naturality condition.

Definition 1.1. Consider functors $F : C \to D$ and $G : C \to D$. A natural transformation $\alpha : F \to G$ (usually depicted as $C \overset{F}{\underset{G}{\Downarrow \alpha}} D$) is a family of morphisms such that

- For each object a in C, α picks a morphism $\alpha_a : F\ a \to G\ a$ in D. α_a is called the component of α at a.
- For each morphism $f : a \to b$ of C the following diagram commutes

$$\begin{array}{ccc} F\ a & \xrightarrow{F\ f} & F\ b \\ {\scriptstyle \alpha_a}\downarrow & & \downarrow{\scriptstyle \alpha_b} \\ G\ a & \xrightarrow{G\ f} & G\ b \end{array}$$

  As an equation $(G\ f) \circ \alpha_a = \alpha_b \circ (F\ f)$.



Definition 1.2. A natural transformation for which each component is an isomorphism is called a *natural isomorphism*. Two functors are isomorphic if there is a natural isomorphism between them.

2 Examples

Here's a slightly non-intuitive example that will go through the pieces of the definition.

Example 2.1. Consider the statement "Every group is naturally isomorphic to its opposite group". Let's dissect and then prove this statement. First we need to know what the opposite group is. Let $(G, *)$ be a group. The opposite group $(G^{op}, *^{op})$ is defined to have $G^{op} = G$ and $a *^{op} b = b * a$. As an exercise check is that $(G^{op}, *^{op})$ is indeed a group.

Consider the category **Grp**, where objects are all groups and morphisms are group homomorphisms. A natural transformation goes between functors. For this example consider the identity

functor on **Grp**, $id_{\mathbf{Grp}} : \mathbf{Grp} \to \mathbf{Grp}$ and the opposite functor $Op : \mathbf{Grp} \to \mathbf{Grp}$. Define the opposite functor as follows: For an object $(G, *)$, $Op\ (G, *) = (G^{op}, *^{op})$. For a morphism f, $Op\ f = f^{op} = f$. Note that for Op to be a functor we must have that f^{op} is a homomorphism from $(G^{op}, *_G^{op})$ to $(H^{op}, *_H^{op})$ for each homomorphism $f : (G, *_G) \to (H, *_H)$. Let a and b be elements of G^{op}

$$f^{op}(a *_G^{op} b) = f(a *_G^{op} b) = f(b *_G a) = f(b) *_H f(b) = f^{op}(a) *_H^{op} f^{op}(b)$$

This essentially shows that f^{op} is a homomorphism and thus Op is a functor.

To prove the initial statement we need to define a natural isomorphism between $id_{\mathbf{Grp}}$ and Op. That is we need an isomorphism η_G for every group G such that the following diagram commutes:

$$\begin{array}{ccc} (G, *_G) = id_{\mathbf{Grp}}\ (G, *_G) & \xrightarrow{f = id_{\mathbf{Grp}}\ f} & (H, *_H) = id_{\mathbf{Grp}}\ (H, *_H) \\ \Big\downarrow{\scriptstyle \eta_G} & & \Big\downarrow{\scriptstyle \eta_H} \\ (G^{op}, *_G^{op}) & \xrightarrow{f^{op} = f} & (H^{op}, *_H^{op}) \end{array}$$

Let $\eta_G(a) = a^{-1}$, where a is an element of $(G, *_G)$ and the rhs is an element of $(G^{op}, *_G^{op})$.

η_G is obviously an isomorphism. It is its own inverse. We need to check two more things. For η_G to be a natural transformation we need to make sure that η_G is a morphism of **Grp** and also check the naturality square.

Is η_G a morphism of **Grp**, in other words, is η_G a group homomorphism?

$$\eta_G(a *_G b) = (a *_G b)^{-1} = b^{-1} *_G a^{-1} = a^{-1} *_G^{op} b^{-1} = \eta_G(a) *_G^{op} \eta_G(b)$$

Thus η_G is a group homomorphism.

Let's check naturality. Consider a group homomorphism $f : (G, *_G) \to (H, *_H)$. For all $a \in G$ we have

$$(\eta_H \circ f)(a) = (f(a))^{-1} = f(a^{-1}) = f^{op}(a^{-1}) = (f^{op} \circ \eta_G)(a)$$

Thus naturality holds.

All told, we have shown that $id_{\mathbf{Grp}}$ is naturally isomorphic to Op. This is saying two things. One, each group is isomorphic to its opposite group. This is due to the existence of an isomorphism between a group and its opposite group. Also, this isomorphism also respects all structure preserving maps. That's what naturality means.

2.1 Natural Transformations in Programming

What would a natural transformation in **Hask** be? In this category functors are endofunctors from types to types. Consider functors $F : \mathbf{Hask} \to \mathbf{Hask}$ and $G : \mathbf{Hask} \to \mathbf{Hask}$. A natural transformation *alpha* must have a component, in this case a haskell function, at every type a, $alpha_a : F\ a \to G\ a$. In programming we often drop the subscript and allow a to vary freely.

Such a function $alpha : F\ a \to G\ a$ is called *polymorphic*. In functional programming there are usually two types of polymorphism: *ad-hoc polymorphism* and *parametric polymorphism*. Without getting too much into it, a parametrically polymorphic function has a single implementation for all types. In contrast, an ad-hoc polymorphic function can have different implementations depending on the incoming type. Operator overloading is an example of ad-hoc polymorphism. Integers could

be handled differently than floats for example. For these notes, we are going to only be using parametrically polymorphic functions.

For a polymorphic function $alpha : F\ a \to G\ a$ to be a natural transformation, naturality must be satisfied. I'll write the condition in haskell syntax.

```
(fmap f) . alpha = alpha . (fmap f)
```

In the above code `f` would have type `a->b`. The rhs `fmap` has type `F a -> F b` and the lhs `fmap` has type `G a -> G b`.

Example 2.2. The `safeHead` function is used to get the head of a possibly empty list. If the list is empty `safeHead` returns `Nothing`.

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

Let's verify naturality. We have two cases depending on whether the incoming list is empty or not

```
(fmap f . safeHead) [] = fmap f Nothing = Nothing
(safeHead. fmap f) [] = safeHead [] = Nothing
```

And if the list is not empty

```
(fmap f . safeHead) (x:xs) = fmap f (Just x) = Just (f x)
(safeHead. fmap f) (x:xs) = safeHead (f x : fmap f xs) = Just (f x)
```

This shows `safeHead` is a natural transformation.

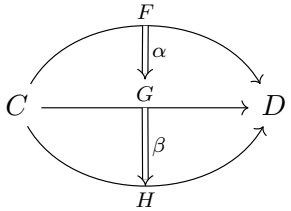
If we think of functors as containers, then a natural transformation modifies the container but not the contents. You can open up the container and rearrange the contents, but you can't modify the contents.

Here's a neat fact about haskell. In the previous example we verified naturality for `safeHead`; however, it turns out if `F` and `G` are really functors, i.e. unit and composition is satisfied, then *any* parametrically polymorphic function with type `F a -> G a` will necessarily be a natural transformation. In actuality, we didn't have to verify naturality for `safeHead`. Because it's a parametrically polymorphic function, the type-checker guarantees naturality.

3 Operations on Natural Transformations

3.1 Vertical Composition

Consider categories C and D and three functors $F : C \to D$, $G : C \to D$, and $H : C \to D$. Let $\alpha : F \to G$ and $\beta : G \to H$ be natural transformations.

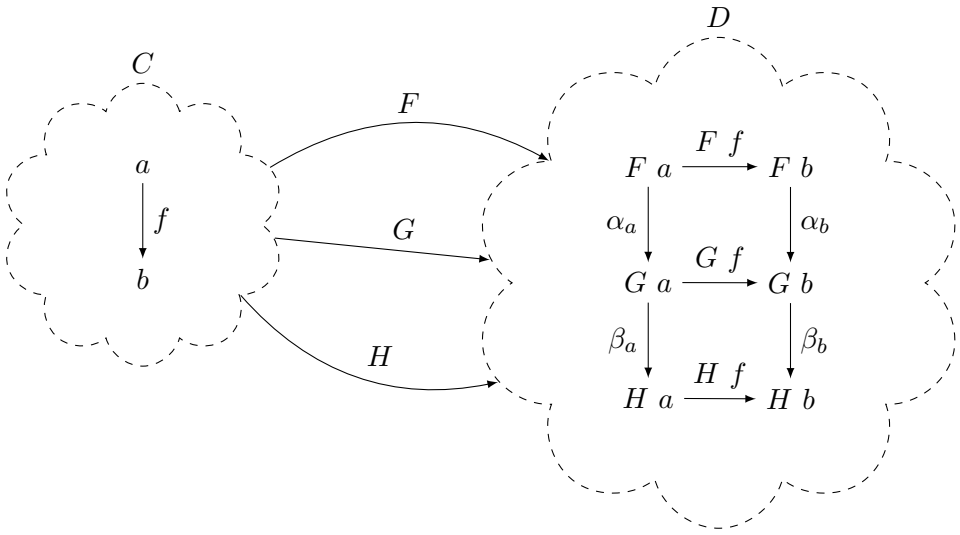


Is there a natural transformation between F and H. Let's consider the components of α and β at a. $\alpha_a : F\ a \to G\ a$ and $\beta_a : G\ a \to H\ a$. Because these are morphisms in D and D is a category

we can compose these arrows for any a of C. We call this composition *vertical* composition, and denote with a ..

$$(\beta.\alpha)_a = \beta_a \circ \alpha_a : F\ a \to H\ a$$

The $\circ$ in the above equation is composition in D. Now we need to check naturality to see if $\beta.\alpha$ is a natural transformation.



The top square commutes in D because α is a natural transformation. Similarly, the bottom square commutes because β is a natural transformation. If all inner diagrams commute, then the whole diagram commutes. Thus,

$$(H\ f) \circ (\beta.\alpha)_a = (\beta.\alpha)_b \circ (F\ f)$$

for every f. This shows $\beta.\alpha$ is a natural transformation.

Observe, we have functors, transformations between functors, and a notion of composition of those transformations. Do we have a category where objects are functors between C and D, and morphisms are natural transformations?

- We have defined composition
- Composition is associative because composition in D is associative.
- We just need identity transformations.

Let $1_F : F \to F$ be a natural transformation defined by $(1_F)_a = id_{Fa}$ for each a of C.

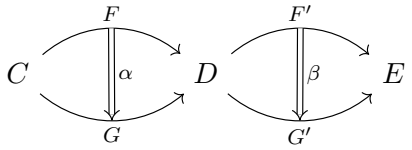
Definition 3.1. Let C and D be categories. The *functor category*, denoted $[C, D]$, $Fun(C, D)$ or D^C, is the category where

- Objects are functors $F : C \to D$.
- Morphisms are natural transformations.
- Composition is vertical composition.
- Identity morphisms are identity natural transformations, 1_F.

The category $[C, C]$ is also called the category of endofunctors.

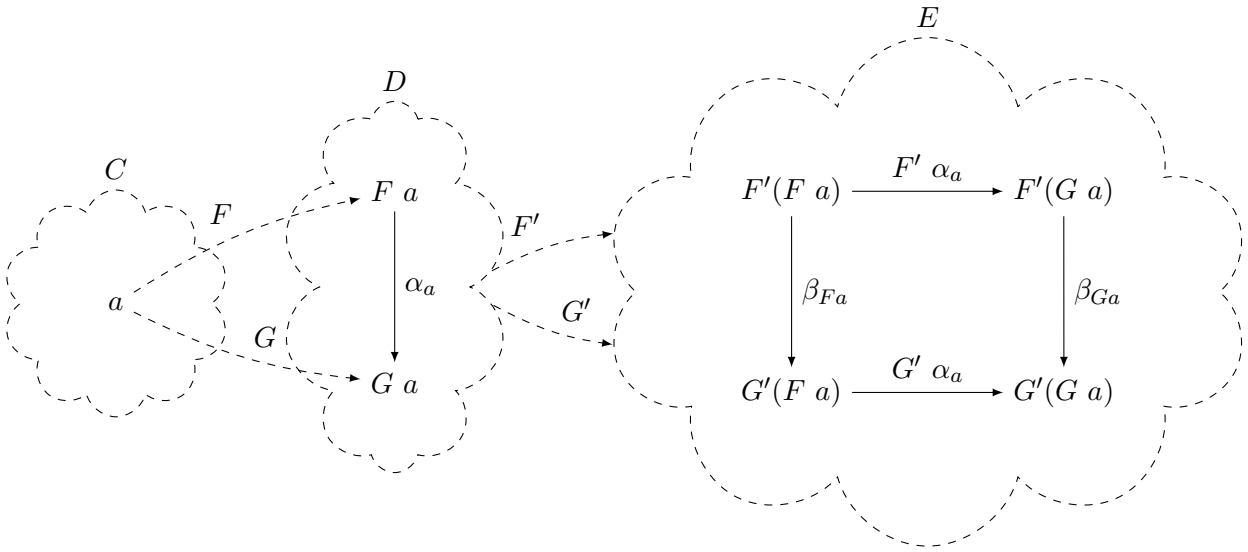
3.2 Horizontal Composition

We wouldn't have called it vertical composition if there was no other kind of composition. Consider the following situation



We can compose functors. Is there a relationship between $F' \circ F : C \to E$ and $G' \circ G : C \to E$? Yes, $\beta \circ \alpha$, called horizontal composition is a natural transformation between $G \circ F$ and $G' \circ F'$.

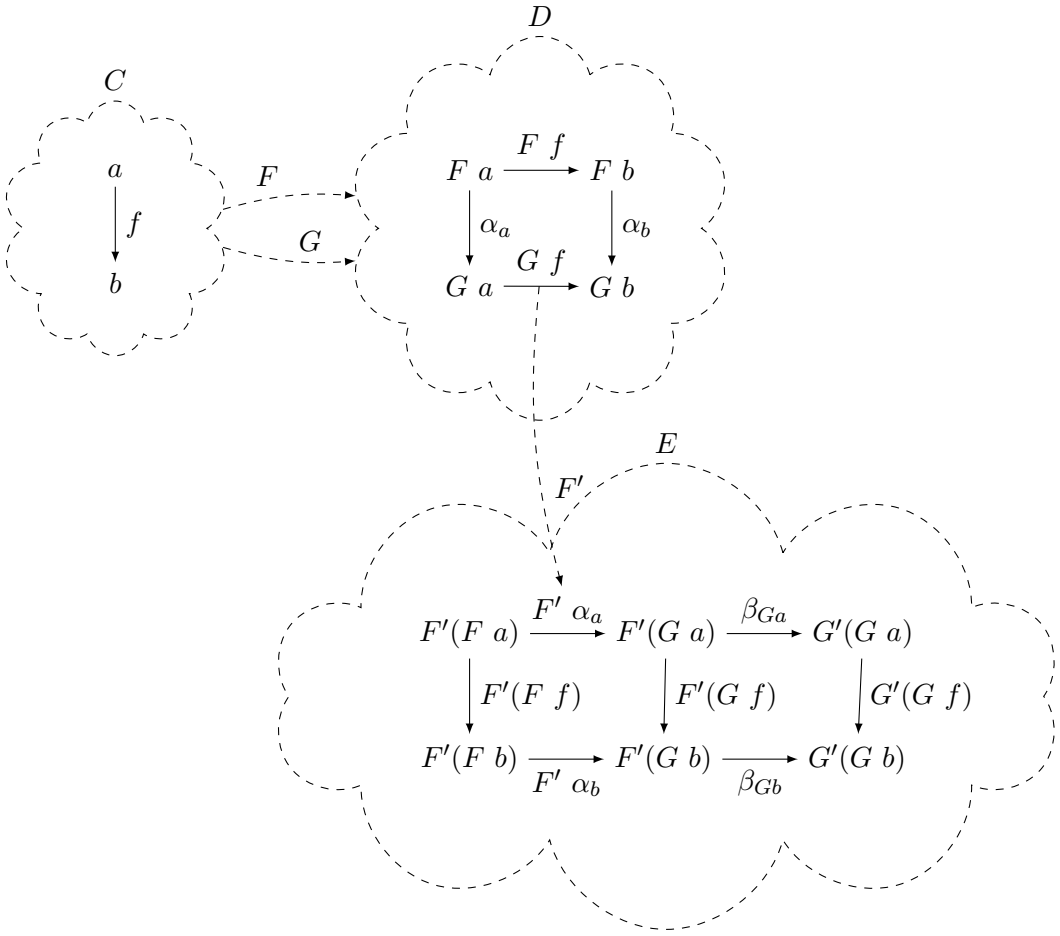
Let's construct components of $\beta \circ \alpha$ and then check naturality. First, note the signature of the relevant component, $(\beta \circ \alpha)_a : (F' \circ F)(a) \to (G' \circ G)(a)$. Now consider the following illustration



The square in E commutes because β is a natural transformation. The commuting square in E thus gives us two ways to define a component of $\beta \circ \alpha$.

$$(\beta \circ \alpha)_a = \beta_{Ga} \circ (F'\ \alpha_a) = (G'\ \alpha_a) \circ \beta_{Fa}$$

Now to check naturality.



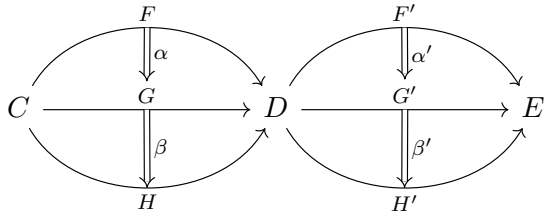
All the diagrams in the above illustration commute. The square in D commutes because α is a natural transformation. The left square of E commutes because F' is a functor and functors preserve commuting diagrams. The right square of E commutes because β is a natural transformation. Reading along the edges of the diagram in E we have

$$(G' \circ G)\ f \circ (\beta \circ \alpha)_a = (G' \circ G)\ f \circ \beta_{Ga} \circ F'\ \alpha_a = \beta_{Gb} \circ F'\ \alpha_b \circ (F' \circ F)\ f = (\beta \circ \alpha)_a \circ (F' \circ F)\ f$$

This is the naturality condition. This shows $\beta \circ \alpha$ is a natural transformation.

4 Interchange Law

Now consider this final situation



$$\beta.\alpha : F \to H \qquad\qquad \alpha' \circ \alpha : F'\ F \to G'\ G$$
$$\beta'.\alpha' : F' \to H' \qquad\qquad \beta' \circ \beta : G'\ G \to H'\ H$$

We now have two different ways to construct a natural transformation from $F'\ F$ to $H'\ H$: $(\beta'.\alpha') \circ (\beta.\alpha)$ and $(\beta' \circ \beta).(\alpha' \circ \alpha)$

Theorem 4.1. *(Interchange law). Take the situation as depicted in the previous illustration*

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

Proof. The following diagram commutes because α' is a natural transformation.

$$\begin{array}{ccccc}
F'\ F\ c & \xrightarrow{F'\ \alpha_c} & F'\ G\ c & \xrightarrow{F'\ \beta_c} & F'\ H\ c \\
 & & \downarrow{\scriptstyle \alpha'_{Gc}} & & \downarrow{\scriptstyle \alpha'_{Hc}} \\
 & & G'\ G\ c & \xrightarrow{G'\ \beta_c} & G'\ H\ c \\
 & & & & \downarrow{\scriptstyle \beta'_{Hc}} \\
 & & & & F'\ H\ c
\end{array}$$

□

$$\beta'_{Hc} \circ \alpha'_{Hc} \circ F'\ \beta_c \circ F'\ \alpha_c = \beta'_{Hc} \circ G'\ \beta_c \circ \alpha'_{Gc} \circ F'\ \alpha_c$$
$$(\beta'.\alpha')_{Hc} \circ F'\ ((\beta.\alpha)_c) = (\beta' \circ \beta)_c \circ (\alpha' \circ \alpha)_c$$
$$((\beta'.\alpha') \circ (\beta.\alpha))_c = ((\beta' \circ \beta).(\alpha' \circ \alpha))_c$$