

## INTRODUCTION

Recall that a functor  $F: \mathcal{C} \rightarrow \mathcal{D}$  is a total function  $F$  that maps the objects and arrows of a category  $\mathcal{C}$  to those of a category  $\mathcal{D}$ , with

$$Ff : FA \rightarrow FB \text{ for each } f : A \rightarrow B$$

and such that the equations

$$F(id_A) = id_{FA} \quad F(g \circ f) = Fg \circ Ff$$

hold for all  $A$  and all  $f, g$  of appropriate types.

Examples: ( $\text{SET} \rightarrow \text{SET}$ )

$P$  = the powerset functor;

List = the list functor;

Tree = the tree functor.

Special cases:

Functors on pre-ordered sets = monotonic functions;

Functors on monoids = monoid homomorphisms;

Functors from product categories = bifunctors.

The category of categories:

$CAT$  = categories as objects, functors as arrows.

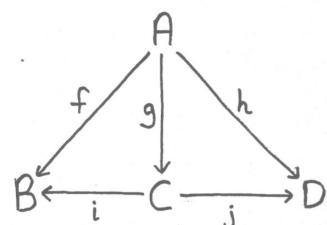
This lecture:

Natural transformations = mappings between functors

## COMMUTING DIAGRAMS

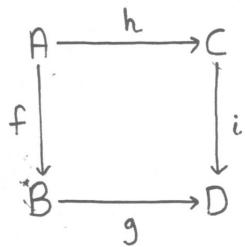
Picturing fragments of a category in a diagram is a useful way of recording type information about particular arrows, i.e. their source/target objects.

Example:



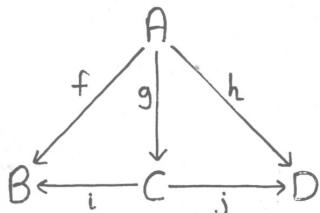
Diagrams are also useful for recording equations about arrows. A diagram commutes when any two paths between objects are equal as composite arrows.

Example: for the diagram



to commute means  $g \circ f = i \circ h$ .

Example: for the diagram



to commute means  $f = i \circ g$  and  $h = j \circ g$ .

## NATURAL TRANSFORMATIONS

Suppose that  $F, G: \mathbb{C} \rightarrow \mathbb{D}$  are two functors.

Then a natural transformation

$$\alpha : F \rightarrow G$$

from  $F$  to  $G$  is a total function  $\alpha$  that maps objects of  $\mathbb{C}$  to arrows of  $\mathbb{D}$ , such that:

① For each object  $A$  in  $\mathbb{C}$ , there is an arrow  $\alpha_A : FA \rightarrow GA$  in  $\mathbb{D}$ .

② For each arrow  $f : A \rightarrow B$  in  $\mathbb{C}$ , the following square commutes in  $\mathbb{D}$ :

$$\begin{array}{ccc}
 FA & \xrightarrow{\alpha_A} & GA \\
 Ff \downarrow & & \downarrow Gf \\
 FB & \xrightarrow{\alpha_B} & GB
 \end{array}$$

## Notes:

- An exception to the general rule is that parallel pairs  $A \rightrightarrows B$  in commuting diagrams are not required to be equal, e.g.

$$\begin{array}{ccccc}
 & & f & & \\
 A & \xrightarrow{\quad} & B & \xrightarrow{h} & C \\
 & & g & & 
 \end{array}$$

means  $h \circ f = h \circ g$  but not  $f = g$ .

- Most commuting diagrams are made up of one or more squares and triangles.
- In category theory, all diagrams are usually assumed to commute unless stated otherwise.

## Notes:

- We usually write  $\alpha_A$  rather than  $\alpha(A)$  when applying a natural transformation to an object.
- The arrows  $\alpha_A, \alpha_B, \alpha_C, \dots$  are called the components of a natural transformation.
- ① states that  $\alpha$  is a transformation; ② states that the transformation is natural.
- Older textbooks sometimes use the notation  $\alpha : F \dot{\rightarrow} G$  for a natural transformation.
- Rather than trying to motivate or explain the definition of a natural transformation, it is more useful to look at some examples.

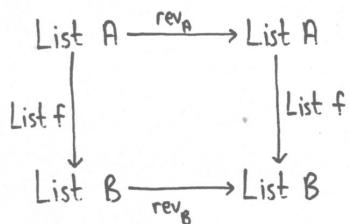
### Example 1: $\text{rev} : \text{List } A \rightarrow \text{List } A$

If  $A$  is a set, then  $\text{rev}_A : \text{List } A \rightarrow \text{List } A$  is the function that reverses a list of elements from  $A$ , and is defined by the equations

$$\text{rev}_A(\text{Nil}) = \text{Nil}$$

$$\text{rev}_A(\text{Cons } x \text{ xs}) = \text{rev}_A(xs) \text{ ++ Cons } x \text{ Nil}$$

In SET, the naturality of  $\text{rev}$



is equivalent to

$$\forall xs. \text{List } f(\text{rev}_A(xs)) = \text{rev}_B(\text{List } f(xs))$$

... which can be verified by induction. The base case  $xs = \text{Nil}$  is trivial, while for the inductive case  $xs = \text{Cons } y \text{ ys}$  we calculate as follows:

$$\begin{aligned} & \text{List } f(\text{rev}_A(\text{Cons } y \text{ ys})) \\ &= \{\text{definition of rev}\} \\ & \text{List } f(\text{rev}_A(ys) \text{ ++ Cons } y \text{ Nil}) \\ &= \{\text{distribution property}\} \\ & \text{List } f(\text{rev}_A(ys)) \text{ ++ List } f(\text{Cons } y \text{ Nil}) \\ &= \{\text{definition of List } f\} \\ & \text{List } f(\text{rev}_A(ys)) \text{ ++ Cons}(f(y)) \text{ Nil} \\ &= \{\text{induction hypothesis}\} \\ & \text{rev}_B(\text{List } f(ys)) \text{ ++ Cons}(f(y)) \text{ Nil} \\ &= \{\text{definition of rev}\} \\ & \text{rev}_B(\text{Cons}(f(y)) (\text{List } f(ys))) \\ &= \{\text{definition of List } f\} \\ & \text{rev}_B(\text{List } f(\text{Cons } y \text{ ys})) \end{aligned}$$

### Notes:

- Naturality of  $\text{rev}$  states that the definition of  $\text{rev}_A$  does not depend on  $A$ . That is, naturality formalises that reversing a list is polymorphic in the type of the elements.
- In Haskell,  $\text{rev}_A$  is written as `reverse`, i.e. the type argument  $A$  is left implicit.
- The concatenation operator  $\text{++}$  is defined by

$$\text{Nil} \text{ ++ ys} = ys$$

$$(\text{Cons } x \text{ xs}) \text{ ++ ys} = \text{Cons } x \text{ (xs ++ ys)}$$

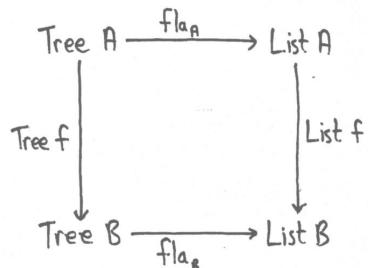
and the distribution of  $\text{List } f$  over  $\text{++}$  can be verified by a simple inductive proof.

### Example 2: $\text{fla} : \text{Tree } A \rightarrow \text{List } A$

If  $A$  is a set, then  $\text{fla}_A : \text{Tree } A \rightarrow \text{List } A$  is the function that flattens a tree of elements from  $A$  to a list, and is defined by the equations

$$\begin{aligned} \text{fla}_A(\text{Leaf } x) &= \text{Cons } x \text{ Nil} \\ \text{fla}_A(\text{Node } l \ r) &= \text{fla}_A(l) \text{ ++ fla}_A(r) \end{aligned}$$

The naturality of  $\text{fla}$



formalises that flattening a tree to a list is polymorphic in the type of the elements.

Example 3:  $\text{len} : \text{List} \rightarrow \text{Int}$

If  $A$  is a set, then  $\text{len}_A : \text{List } A \rightarrow \mathbb{Z}$  is the function that calculates the length of a list of elements from  $A$ , and is defined by the equations

$$\begin{aligned}\text{len}_A(\text{Nil}) &= 0 \\ \text{len}_A(\text{Cons } x : xs) &= 1 + \text{len}_A(xs)\end{aligned}$$

To view the functions  $\text{len}_A$  as the components of a natural transformation, we define a constant functor  $\text{Int} : \text{SET} \rightarrow \text{SET}$  as follows:

Objects -  $\text{Int } A = \mathbb{Z}$

Arrows -  $\text{Int } f = \text{id}_{\mathbb{Z}}$

Hence, we now have  $\text{len}_A : \text{List } A \rightarrow \text{Int } A$

The naturality of  $\text{len}$  simplifies to

$$\begin{array}{ccc}\text{List } A & \xrightarrow{\text{len}_A} & \mathbb{Z} \\ \downarrow \text{List } f & & \downarrow \text{id}_{\mathbb{Z}} \\ \text{List } B & \xrightarrow{\text{len}_B} & \mathbb{Z}\end{array}$$

and formalises that calculating the length of a list is polymorphic in the type of the elements.

More generally:

Many natural transformations used in computing are similar to  $\text{rev}$ ,  $\text{fla}$  and  $\text{len}$ , in that:

$\alpha$  = a polymorphic program

## NATURAL TRANSFORMATIONS AS ARROWS

The notation  $\alpha : F \rightarrow G$  suggests that natural transformations can be viewed as arrows in a category.

Definition: if  $C$  and  $D$  are categories, then the functor category  $D^C$  is defined as follows:

Objects -  $F : C \rightarrow D, \dots$  are functors.

Arrows -  $\alpha : F \rightarrow G, \dots$  are natural transformations.

Identities -  $\text{id}_F : F \rightarrow F$  is the identity natural transformation, defined by  $(\text{id}_F)_A = \text{id}_{FA}$ .

Composition - if  $\alpha : F \rightarrow G$  and  $\beta : G \rightarrow H$  then  $\beta \circ \alpha : F \rightarrow H$  is the composite natural transformation, defined by  $(\beta \circ \alpha)_A = \beta_A \circ \alpha_A$ .

3.13

3.14

The naturality of  $\beta \circ \alpha$

$$\begin{array}{ccc}FA & \xrightarrow{(\beta \circ \alpha)_A} & HA \\ \downarrow FF & & \downarrow HF \\ FB & \xrightarrow{(\beta \circ \alpha)_B} & HB\end{array}$$

can be verified by the following calculation:

$$HF \circ (\beta \circ \alpha)_A$$

= {definition of  $\circ$ }

$$HF \circ \beta_A \circ \alpha_A$$

= {naturality of  $\beta$ }

$$\beta_B \circ GF \circ \alpha_A$$

= {naturality of  $\alpha$ }

$$\beta_B \circ \alpha_B \circ FF$$

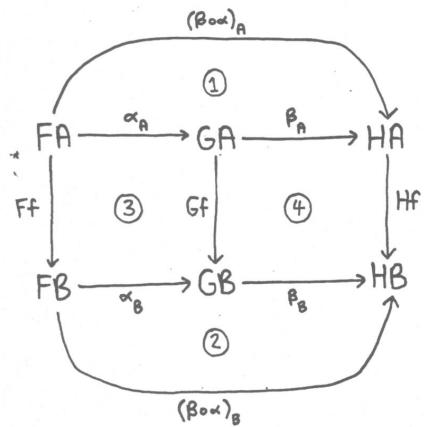
= {definition of  $\circ$ }

$$(\beta \circ \alpha)_B \circ FF$$

3.15

3.16

Equivalently, the naturality of  $\beta \circ \alpha$  can be verified pictorially. Consider the following diagram:



Then it is easy to see that because

- ① and ② commute (by definition)
- ③ and ④ commute (by assumption)

the outer rectangle also commutes, as required.

## Notes

- Pictorial proof ("diagram chasing") is better suited to a whiteboard than to a written document.
- The naturality of  $\beta \circ \alpha$  formalises that polymorphic programs are closed under composition, e.g.  $\text{len} \circ \text{fla} : \text{Tree} \rightarrow \text{Int}$  is the polymorphic program that calculates the number of leaves in a tree.

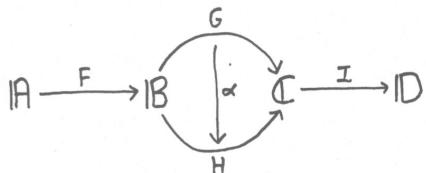
- The naturality of  $\text{id}_F$  can be verified similarly, and formalises that the identity program for any functor  $F$  is polymorphic.

## THE GODEMENT CALCULUS

We conclude by reviewing some other useful notions of composition for natural transformations, first studied by Godelement in 1958.

### Composition with functors:

Suppose that we are given a diagram



where

$IA, IB, IC, ID$  are categories;

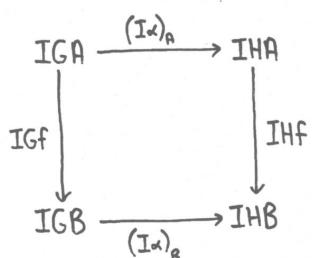
$F, G, H, I$  are functors;

$\alpha$  is a natural transformation.

Then:

- Precomposing  $\alpha$  with the functor  $F$  gives a natural transformation  $\alpha_F : GF \rightarrow HF$ , defined by  $(\alpha_F)_A = \alpha_{(FA)}$ ;
- Postcomposing  $\alpha$  with the functor  $I$  gives a natural transformation  $I\alpha : IG \rightarrow IH$ , defined by  $(I\alpha)_A = I(\alpha_A)$ .

For example, the naturality of  $I\alpha$



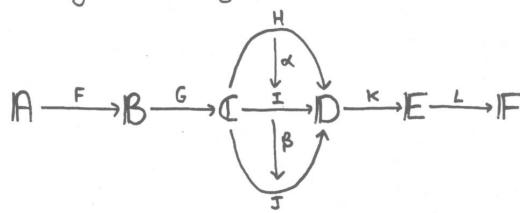
can be verified as follows ...

$$\begin{aligned}
 & \text{IHf} \circ (\text{I}\alpha)_A \\
 = & \{ \text{definition of } \text{I}\alpha \} \\
 & \text{IHf} \circ \text{I}(\alpha_A) \\
 = & \{ \text{I is a functor} \} \\
 & \text{I}(\text{Hf} \circ \alpha_A) \\
 \Rightarrow & \{ \text{naturality of } \alpha \} \\
 & \text{I}(\alpha_B \circ \text{Gf}) \\
 = & \{ \text{I is a functor} \} \\
 & \text{I}(\alpha_B) \circ \text{IGf} \\
 = & \{ \text{definition of } \text{I}\alpha \} \\
 & (\text{I}\alpha)_B \circ \text{IGf}
 \end{aligned}$$

### Note:

The choice of notation allows terms of the form  $\alpha_{FA}$  and  $\text{I}\alpha_A$  to be written without parentheses, because  $(\alpha_F)_A = \alpha_{(FA)}$  and  $(\text{I}\alpha)_A = \text{I}(\alpha_A)$ .

Now, given a diagram



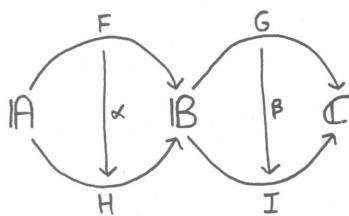
we have the following equational laws:

- ①  $\alpha_{(GF)} = (\alpha_G)_F$
- ②  $(LK)\alpha = L(K\alpha)$
- ③  $(K\alpha)_G = K(\alpha_G)$
- ④  $K(\beta \circ \alpha)_G = KB_G \circ K\alpha_G$

Note: equations ①-③ again allow terms of certain forms to be written without parentheses.

### Horizontal composition:

Suppose that we are given a diagram

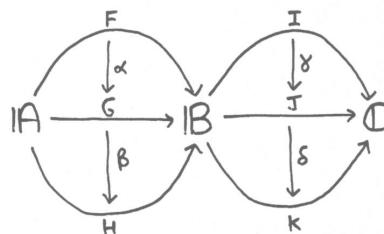


Then composing  $\alpha$  and  $\beta$  gives a natural transformation  $\beta * \alpha : GF \rightarrow IH$ , defined by  $\beta * \alpha = I\alpha \circ \beta_F = \beta_H \circ G\alpha$ .

Note: on natural transformations,

- o is called vertical composition;
- \* is called horizontal composition.

Now, given a diagram



we have the following interchange law that relates vertical and horizontal composition:

$$⑤ (\delta * \gamma) * (\beta * \alpha) = (\delta * \beta) \circ (\gamma * \alpha)$$

### Note:

Verifying equations ①-⑤ is an excellent way to become familiar with natural transformations.