Introduction to Category Theory

Graham Hutton

Lecture 3 - Natural Transformations

INTRODUCTION

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

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

and such that the equations

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

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

Examples: (**SET** → **SET**)

$\mathbb{P}$ = 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:

$\boxed{\text{Natural transformations = mappings between functors}}$

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:

$$\begin{tikzcd} & A \arrow[dl, "f"'] \arrow[d, "g"] \arrow[dr, "h"] & \\ B & C \arrow[l, "i"] \arrow[r, "j"'] & D \end{tikzcd}$$

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

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

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

Example: for the diagram

$$\begin{tikzcd} & A \arrow[dl, "f"'] \arrow[d, "g"] \arrow[dr, "h"] & \\ B & C \arrow[l, "i"] \arrow[r, "j"'] & D \end{tikzcd}$$

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

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.

$$\begin{tikzcd} A \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & B \arrow[r, "h"] & C \end{tikzcd}$$

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.

NATURAL TRANSFORMATIONS

Suppose that $F, G : \mathbb{C} \to \mathbb{D}$ are two functors. Then a natural transformation

$$\alpha : F \to G$$

from F to G is a total function α 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 $\alpha_A : FA \to GA$ in $\mathbb{D}$.

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

$$\begin{tikzcd} FA \arrow[r, "\alpha_A"] \arrow[d, "Ff"'] & GA \arrow[d, "Gf"] \\ FB \arrow[r, "\alpha_B"'] & GB \end{tikzcd}$$

Notes:

- We usually write α_A rather than $\alpha(A)$ when applying a natural transformation to an object.

- The arrows $\alpha_A, \alpha_B, \alpha_C, \ldots$ are called the components of a natural transformation.

- ① states that α is a transformation; ② states that the transformation is natural.

- Older textbooks sometimes use the notation $\alpha : F \dot{\to} G$ 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: rev : List → List

If A is a set, then $rev_A : List\ A \to List\ A$ is the function that reverses a list of elements from A, and is defined by the equations

$$rev_A\ (Nil) = Nil$$
$$rev_A\ (Cons\ x\ xs) = rev_A\ (xs) +\!\!+ Cons\ x\ Nil$$

In **SET**, the naturality of rev

$$\begin{tikzcd} List\ A \arrow[r, "rev_A"] \arrow[d, "List\ f"'] & List\ A \arrow[d, "List\ f"] \\ List\ B \arrow[r, "rev_B"'] & List\ B \end{tikzcd}$$

is equivalent to

$$\forall xs.\ List\ f\ (rev_A\ (xs)) = rev_B\ (List\ f\ (xs))$$

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

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

Notes:

- Naturality of rev states that the definition of 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, rev_A is written as reverse, i.e. the type argument A is left implicit.
- The concatenation operator $+\!\!+$ is defined by

$$Nil +\!\!+ ys = ys$$
$$(Cons\ x\ xs) +\!\!+ ys = Cons\ x\ (xs +\!\!+ ys)$$

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

Example 2: fla : Tree → List

If A is a set, then $fla_A : Tree\ A \to List\ A$ is the function that flattens a tree of elements from A to a list, and is defined by the equations

$$fla_A\ (Leaf\ x) = Cons\ x\ Nil$$
$$fla_A\ (Node\ l\ r) = fla_A\ (l) +\!\!+ fla_A\ (r)$$

The naturality of fla

$$\begin{tikzcd} Tree\ A \arrow[r, "fla_A"] \arrow[d, "Tree\ f"'] & List\ A \arrow[d, "List\ f"] \\ Tree\ B \arrow[r, "fla_B"'] & List\ B \end{tikzcd}$$

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

Example 3: len : List → Int

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

$$len_A\ (Nil) = 0$$
$$len_A\ (Cons\ x\ xs) = 1 + len_A\ (xs)$$

To view the functions len_A as the components of a natural transformation, we define a constant functor $Int : \mathbf{SET} \to \mathbf{SET}$ as follows:

Objects – $Int\ A = \mathbb{Z}$

Arrows – $Int\ f = id_{\mathbb{Z}}$

Hence, we now have $len_A : List\ A \to Int\ A$

The naturality of len simplifies to

$$\begin{tikzcd} List\ A \arrow[r, "len_A"] \arrow[d, "List\ f"'] & \mathbb{Z} \arrow[d, "id_{\mathbb{Z}}"] \\ List\ B \arrow[r, "len_B"'] & \mathbb{Z} \end{tikzcd}$$

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 rev, fla and len, in that:
>
> α = a polymorphic program

NATURAL TRANSFORMATIONS AS ARROWS

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

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

Objects – $F : \mathbb{C} \to \mathbb{D}, \ldots$ are functors.

Arrows – $\alpha : F \to G, \ldots$ are natural transformations.

Identities – $id_F : F \to F$ is the identity natural transformation, defined by $(id_F)_A = id_{FA}$.

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

The naturality of $\beta \circ \alpha$

$$\begin{tikzcd} FA \arrow[r, "(\beta\circ\alpha)_A"] \arrow[d, "Ff"'] & HA \arrow[d, "Hf"] \\ FB \arrow[r, "(\beta\circ\alpha)_B"'] & HB \end{tikzcd}$$

can be verified by the following calculation:

$$\begin{aligned} & Hf \circ (\beta \circ \alpha)_A \\ = & \quad \{\text{definition of } \circ\} \\ & Hf \circ \beta_A \circ \alpha_A \\ = & \quad \{\text{naturality of } \beta\} \\ & \beta_B \circ Gf \circ \alpha_A \\ = & \quad \{\text{naturality of } \alpha\} \\ & \beta_B \circ \alpha_B \circ Ff \\ = & \quad \{\text{definition of } \circ\} \\ & (\beta \circ \alpha)_B \circ Ff \end{aligned}$$

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

$$\begin{tikzcd}
FA \arrow[r, "\alpha_A"] \arrow[d, "Ff"'] \arrow[rr, bend left, "(\beta\circ\alpha)_A"] \arrow[dr, phantom, "③"] & GA \arrow[r, "\beta_A"] \arrow[d, "Gf"] \arrow[dr, phantom, "①"] \arrow[dr, phantom, "④"] & HA \arrow[d, "Hf"] \\
FB \arrow[r, "\alpha_B"'] \arrow[rr, bend right, "(\beta\circ\alpha)_B"'] \arrow[rr, phantom, shift right=3ex, "②"] & GB \arrow[r, "\beta_B"'] & HB
\end{tikzcd}$$

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.

  $\mathrm{len} \circ \mathrm{fla} : \mathrm{Tree} \to \mathrm{Int}$ is the polymorphic program that calculates the number of leaves in a tree.

- The naturality of 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 Godement in 1958.

Composition with functors:

Suppose that we are given a diagram

$$\begin{tikzcd}
\mathbb{A} \arrow[r, "F"] & \mathbb{B} \arrow[r, bend left, "G"] \arrow[r, bend right, "H"'] \arrow[r, phantom, "\Downarrow\alpha"] & \mathbb{C} \arrow[r, "I"] & \mathbb{D}
\end{tikzcd}$$

where

$\mathbb{A}, \mathbb{B}, \mathbb{C}, \mathbb{D}$ are categories;

F, G, H, I are functors;

α is a natural transformation.

Then:

- Precomposing α with the functor F gives a natural transformation $\alpha_F : GF \to HF$, defined by $(\alpha_F)_A = \alpha_{(FA)}$;

- Postcomposing α with the functor I gives a natural transformation $I\alpha : IG \to IH$, defined by $(I\alpha)_A = I(\alpha_A)$.

For example, the naturality of $I\alpha$

$$\begin{tikzcd}
IGA \arrow[r, "(I\alpha)_A"] \arrow[d, "IGf"'] & IHA \arrow[d, "IHf"] \\
IGB \arrow[r, "(I\alpha)_B"'] & IHB
\end{tikzcd}$$

can be verified as follows ...

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

<u>Note:</u>

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

Now, given a diagram

$$
\begin{tikzcd}
\mathbb{A} \arrow[r, "F"] & \mathbb{B} \arrow[r, "G"] & \mathbb{C} \arrow[r, bend left=60, "H", "\alpha"'] \arrow[r, "I"] \arrow[r, bend right=60, "J"', "\beta"] & \mathbb{D} \arrow[r, "K"] & \mathbb{E} \arrow[r, "L"] & \mathbb{F}
\end{tikzcd}
$$

we have the following <u>equational laws</u>:

① $\alpha_{(GF)} = (\alpha_G)_F$

② $(LK)\alpha = L(K\alpha)$

③ $(K\alpha)_G = K(\alpha_G)$

④ $K(\beta \circ \alpha)_G = K\beta_G \circ K\alpha_G$

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

<u>Horizontal composition:</u>

Suppose that we are given a diagram

$$
\begin{tikzcd}
\mathbb{A} \arrow[r, bend left=60, "F", "\alpha"'] \arrow[r, bend right=60, "H"'] & \mathbb{B} \arrow[r, bend left=60, "G", "\beta"'] \arrow[r, bend right=60, "I"'] & \mathbb{C}
\end{tikzcd}
$$

Then <u>composing</u> α and β gives a natural transformation $\beta * \alpha : GF \to IH$, defined by

$$\beta * \alpha = I\alpha \circ \beta_F = \beta_H \circ G\alpha.$$

<u>Note</u>: on natural transformations,

- $\circ$ is called <u>vertical</u> composition;
- $*$ is called <u>horizontal</u> composition.

Now, given a diagram

$$
\begin{tikzcd}
\mathbb{A} \arrow[r, bend left=60, "F", "\alpha"'] \arrow[r, "G"] \arrow[r, bend right=60, "H"', "\beta"] & \mathbb{B} \arrow[r, bend left=60, "I", "\gamma"'] \arrow[r, "J"] \arrow[r, bend right=60, "K"', "\delta"] & \mathbb{C}
\end{tikzcd}
$$

we have the following <u>interchange law</u> that relates vertical and horizontal composition:

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

<u>Note:</u>

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