On two tricks to make Category Theory fit in less mental space: missing diagrams and skeletons of proofs

A talk at the “Creativity 2019” conference
in honor of Newton da Costa’s 90th birthday
Rio de Janeiro, december 11, 2019

By: Eduardo Ochs

eduardoochs@gmail.com
http://angg.twu.net/math-b.html#2019-newton

Introduction

Most texts in Category Theory ("CT" from here on) are full of expressions like this:

"Let's write $(A\times)$ for the functor that takes each object B to $A \times B$"

I was absolutely fascinated by this "the".
A functor — say, $(A\times)$ — has an action on objects, an action on morphisms, and guarantees, or proofs, that it respects identities and compositions.

That "the functor" implies that the reader should be able to figure out by himself the action on morphisms, i.e., the precise meaning for $(A\times)f$ when $f : B \to C$, and to check that this $(A\times)$ respects identities and compositions.

Introduction (2)

Formally, a functor $(A\times) : \mathbf{Set} \to \mathbf{Set}$
is a 4-uple:

$$(A\times) = ((A\times)_0, (A\times)_1, \mathsf{respids}_{(A\times)}, \mathsf{respcomp}_{(A\times)})$$

The “the” in

“$(A\times)$ is the functor that takes each object B to $A\times B$”

suggests that learning CT transforms you in a certain way...
you become a person who can infer $(A\times)_1$, $\mathsf{respids}_{(A\times)}$,
and $\mathsf{respcomp}_{(A\times)}$ from just $(A\times)_0$...

...you become a person who can define functors in a very compact way, and the other CT people will understand you.

(I wanted to become like that when I'd grow up)

Functions with and without names

Consider this function:

$$\begin{array}{rcrcl} f & : & \{1,2,3\} & \to & \mathbb{Z} \\ & & a & \mapsto & 10a \end{array}$$

It has a name: f.

There are two easy ways to work with functions without names...

Lambda notation

Way 1: A function is a set of input-output pairs:

$f = \{(1,10),(2,20),(3,30)\}$

$$\text{So:}\quad \begin{array}{rcl} f(2) & = & \{(1,10),(2,20),(3,30)\}(2) \\ & = & 20 \end{array}$$

Way 2: A function is a program in λ-notation:

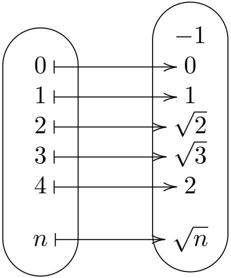
$f = (\lambda a.10a)$

$$\text{So:}\quad \begin{array}{rcl} f(2) & = & (\lambda a.10 \cdot a)(2) \\ & = & (10 \cdot a)[a := 2] \\ & = & 10 \cdot 2 \\ & = & 20 \end{array}$$

Both ways drop some information:
name, codomain, and, in the case of $(\lambda a.10 \cdot a)$, domain.
There is a also this notation: $(\lambda a{:}\{1,2,3\}.10 \cdot a)$, that includes the domain (a "type"!), but we are in a hurry...

Internal diagrams

$$\begin{array}{rccc} \sqrt{\ } : & \mathbb{N} & \to & \mathbb{R} \\ & n & \mapsto & \sqrt{n} \end{array}$$



$$\mathbb{N} \xrightarrow{\sqrt{\ }} \mathbb{R}$$

The $n \longmapsto \sqrt{n}$ shows how $\sqrt{\ }$ acts on a generic element.
The $3 \longmapsto \sqrt{3}$ shows how $\sqrt{\ }$ acts on a particular element.
The $4 \longmapsto 2$ shows how $\sqrt{\ }$ acts on another element.

Internal diagrams in categories

Above: internal view (without the blobs)

Below: external view

$$
\begin{tikzcd}
C \arrow[r, mapsto, "F_0"] \arrow[d, "g"'] & FC \arrow[d, "Fg"] \\
D \arrow[r, mapsto, "F_0"] & FD
\end{tikzcd}
\qquad \overset{F_1}{\longmapsto}
$$

$$
\mathbf{A} \xrightarrow{F} \mathbf{B}
$$

Above $\mathbf{A}$: objects and morphisms of $\mathbf{A}$ (same for $\mathbf{B}$)

Above F: the actions of F on objs and morphisms

(Some conventions come from fibrations)

The shape of Beck-Chevalley

$$
\begin{tikzcd}
 & f'^*P \arrow[dl] \arrow[dr] \arrow[ddr, mapsto] & & P \arrow[ll, mapsto] \arrow[d] \arrow[ddr, mapsto] & \\
z'^*f^*\Sigma_z P \arrow[rr, leftrightarrow] \arrow[drr, mapsto] & & f'^*z^*\Sigma_z P & z^*\Sigma_z P \arrow[l, mapsto] & \\
 & & \Sigma_{z'} f'^*P \arrow[d, shift right, "\natural"'] & & \Sigma_z P \arrow[d, "\mathrm{id}"] \\
 & & f^*\Sigma_z P \arrow[u, shift right, "\mathsf{BCCL}"'] \arrow[uull, mapsto] & & \Sigma_z P \arrow[ll, mapsto] \arrow[uul, mapsto] \\
X \times_Y Z \arrow[rrr, "f'"'] \arrow[drr, "z'"'] \arrow[dr, phantom, "\lrcorner"] & & & Z \arrow[dr, "z"] & \\
 & & X \arrow[rr, "f"] & & Y
\end{tikzcd}
$$

What I was trying to understand
Short answer: categorical semantics should be more intuitive
Part of the long answer: hyperdoctrines are important
but the definition of hyperdoctrine is super-hard...

A hyperdoctrine is a fibration $p : \mathbb{E} \to \mathbb{B}$
over a base category $\mathbb{B}$ with finite products,
in which each fiber is cartesian-closed, and
in which every change-of-base functor f^*
has adjoints $\Sigma_f \dashv f^* \dashv \Pi_f$...
Also, all Beck-Chevalley maps and
all Frobenius maps in it are invertible
(yuck! Plus lots of details...)

What are the intended semantics of these operations?
Can I work in the abstract definition and in the
intended semantics "in parallel"?

Parallel diagrams

...what are the intended semantics of these operations?
Can I work in the abstract definition and in the intended semantics "in parallel"?

Yes, if by "in parallel" we mean
"using diagrams with the same shape".
An example:

$$
\begin{tikzcd}
X \arrow[r, mapsto, "F_0"] \arrow[d, "g"'] \arrow[dr, phantom, "\overset{F_1}{\longmapsto}"] & FX \arrow[d, "Fg"] \\
Y \arrow[r, mapsto, "F_0"] & FY \\
\mathbf{A} \arrow[r, "F"] & \mathbf{B}
\end{tikzcd}
\qquad
\begin{tikzcd}
B \arrow[r, mapsto, "(A\times)_0"] \arrow[d, "f"'] \arrow[dr, phantom, "\overset{(A\times)_1}{\longmapsto}"] & (A\times)B \arrow[d, "(A\times)f"] \\
C \arrow[r, mapsto, "(A\times)_0"] & (A\times)C \\
\mathbf{Set} \arrow[r, "(A\times)"] & \mathbf{Set}
\end{tikzcd}
\qquad
\begin{tikzcd}
B \arrow[r, mapsto, "(A\times)_0"] \arrow[d, "f"'] \arrow[dr, phantom, "\overset{(A\times)_1}{\longmapsto}"] & A\times B \arrow[d, "\lambda p.(\pi p, f(\pi' p))"] \\
C \arrow[r, mapsto, "(A\times)_0"] & A\times C \\
\mathbf{Set} \arrow[r, "(A\times)"] & \mathbf{Set}
\end{tikzcd}
$$

Parallel diagrams - Logic for Children

The main techniques discussed in the workshop "Logic for Children" (in the UniLog 2018, in Vichy) involved parallel diagrams...

$$\left(\begin{array}{c}\text{particular}\\ \text{case}\\ \text{“for children”}\end{array}\right) \underset{\substack{\text{generalize}\\ \text{(hard)}}}{\overset{\substack{\text{particularize}\\ \text{(easy)}}}{\leftrightarrows}} \left(\begin{array}{c}\text{general}\\ \text{case}\\ \text{“for adults”}\end{array}\right)$$

$$\left(\begin{array}{c}\text{intended}\\ \text{meaning}\end{array}\right) \dashleftarrow\!\dashrightarrow \left(\begin{array}{c}\text{categorical}\\ \text{semantics}\end{array}\right)$$

$$\left(\begin{array}{c}\text{internal +}\\ \text{external}\end{array}\right) \dashleftarrow\!\dashrightarrow \left(\begin{array}{c}\text{external}\\ \text{view}\end{array}\right)$$

An example from Topos Theory

In 2018 I was using these techniques —
parallel diagrams, internal views,
particular cases, finite examples "for children"
— to understand things in Topos Theory...

I was super happy because with these techniques
I finally was able to understand some things
about toposes and sheaves, that before were
MUCH more abstract than my brain could handle...

..and I showed this figure, of a particular case
of a geometric morphism that induces a sheaf...

(This particular case is rich enough to give me
a lot of intuition about GMs and shaves)

$$
\left(\begin{tikzcd}[sep=small] F_2 \arrow[dr] & & F_3 \arrow[dl] \arrow[dr] & \\ & F_4 & & F_5 \end{tikzcd}\right)
\longmapsfrom
\left(\begin{tikzcd}[sep=small] & F_1 \arrow[dl] \arrow[dr] & & \\ F_2 \arrow[dr] & & F_3 \arrow[dl] \arrow[dr] & \\ & F_4 \arrow[dr] & & F_5 \arrow[dl] \\ & & F_6 & \end{tikzcd}\right)
$$

$$
\Big\downarrow \qquad\qquad \longleftrightarrow \qquad\qquad \Big\downarrow
$$

$$
\left(\begin{tikzcd}[sep=small] G_2 \arrow[dr] & & G_3 \arrow[dl] \arrow[dr] & \\ & G_4 & & G_5 \end{tikzcd}\right)
\longmapsto
\left(\begin{tikzcd}[sep=small] & G_2 \times_{G_4} G_3 \arrow[dl] \arrow[dr] & & \\ G_2 \arrow[dr] & & G_3 \arrow[dl] \arrow[dr] & \\ & G_4 \arrow[dr] & & G_5 \arrow[dl] \\ & & 1 & \end{tikzcd}\right)
$$

$$
\mathbf{Set}^{\mathbf{A}} \underset{f_*}{\overset{f^*}{\leftrightarrows}} \mathbf{Set}^{\mathbf{B}}
$$

$$
\left(\begin{tikzcd}[sep=small] 2 \arrow[dr] & & 3 \arrow[dl] \arrow[dr] & \\ & 4 & & 5 \end{tikzcd}\right)
\xrightarrow{f}
\left(\begin{tikzcd}[sep=small] & 1 \arrow[dl] \arrow[dr] & & \\ 2 \arrow[dr] & & 3 \arrow[dl] \arrow[dr] & \\ & 4 \arrow[dr] & & 5 \arrow[dl] \\ & & 6 & \end{tikzcd}\right)
$$

(for children; inclusion, sheaf)

$$
\begin{tikzcd} f^*F \arrow[d] \arrow[dr, phantom, "\longleftrightarrow"] & F \arrow[l] \arrow[d] \\ G \arrow[r] & f_*G \end{tikzcd}
$$

$$
\mathcal{F} \underset{f_*}{\overset{f^*}{\leftrightarrows}} \mathcal{E}
$$

(for adults)

I felt that I had some techniques for creating
"the right (finite) examples", and these examples
could give me/us a lot of intuition on Topos Theory...

That was quite nice, but then I started to ask:
what exactly is this "intuition"?
What kinds of knowledge are transferred
between parallel diagrams?

My first answer was:
in two parallel diagrams A and B
with entities $A_1, \ldots, A_n$, $B_1, \ldots, B_n$,
the relations between the entities in A
and the correpondent entities in B
are the same if we see these entities as λ-terms.

So: **let's study this!** $\uparrow$

Formalizing the Yoneda Lemma in λ-calculus
See: http://angg.twu.net/math-b.html#notes-yoneda

$$\begin{tikzcd}
& 1 \arrow[d, "\gamma"] \\
C \arrow[r, mapsto] & RC \\
(C{\to}_) \arrow[r, "T"'] \arrow[dr, "T'"'] & (1{\to}R_) \arrow[d, leftrightarrow] \\
& R
\end{tikzcd}
\qquad
\begin{tikzcd}
& 1 \arrow[d, "\gamma"] \\
C \arrow[r, mapsto] & (B{\to}C) \\
(C{\to}_) \arrow[r, "T"'] \arrow[dr, "T'"'] & (1{\to}(B{\to}_)) \arrow[d, leftrightarrow] \\
& (B{\to}_)
\end{tikzcd}$$

Formalizing the Yoneda Lemma in λ-calculus (2)

$$
\begin{tikzcd}
 & A \arrow[d, "\gamma"] \\
C \arrow[r, mapsto] & RC
\end{tikzcd}
$$

$$
(C{\to}\_) \xrightarrow{T} (A{\to}R\_)
$$

$A \in \mathbf{A}$
$C \in \mathbf{C}$
$R : \mathbf{A} \to \mathbf{C}$
$\gamma : A \to RC$
$\gamma := TC(\mathrm{id}_C)$

$(C{\to}\_) : \mathbf{C} \to \mathbf{Set}$
$(C{\to}\_)_0(D) = \mathrm{Hom}_{\mathbf{C}}(C, D)$
$(C{\to}\_)_1(h) = \lambda g.(g; h)$

$(A{\to}R\_) : \mathbf{C} \to \mathbf{Set}$
$(A{\to}R\_)_0(D) = \mathrm{Hom}_{\mathbf{A}}(A, RD)$
$(A{\to}R\_)_1(h) = \lambda \delta.(\delta; Rh)$

$T : (C{\to}\_) \to (A{\to}R\_)$

Categories, functors, NTs, etc, in Idris-ct

I decided to grow up,
and instead of only writing the formalizations
of my diagrams as λ-terms "by hand"
in a system of λ-calculus with dependent types,
as I've been doing for ages —

I would finally learn a language with dependent types
that doubles as a proof assistant: Idris —
and I would implement my Yoneda — or at least
its translation to λ-terms — in Idris, on top
of its library for Category Theory, Idris-ct...

Skeletons (1)
Remember that a functor $(A\times) : \mathbf{Set} \to \mathbf{Set}$
is a 4-uple:

$$(A\times) = ((A\times)_0, (A\times)_1; \mathsf{respids}_{(A\times)}, \mathsf{respcomp}_{(A\times)})$$

The components before the ';'
don't mention equalities of morphisms,
the components after the ';' do.
If we drop the components after the ';' we get

$$(A\times) = ((A\times)_0, (A\times)_1)$$

A "proto-functor".
It is possible to do something similar for
(proto)categories, (proto)isos, (proto)NTs,
(proto)adjunctions, (proto)fibrations,
(proto)hyperdoctrines...

Skeletons (1)
Most constructions and proofs in Category Theory can be done first on the proto-things and then "lifted" to the real things.

The constructions with only the proto-parts are easier and very visual, and they work as "skeletons" for the real constructions and proofs.

I published this idea in a paper in Logica Universalis, "Internal Diagrams and Archetypal Reasoning in Category Theory" (2013), but no one paid any attention. (Link: `http://angg.twu.net/math-b.html#idarct`)

I created a modified version of Idris-ct that defines protocats, protofunctors, etc, instead of cats, functors, etc, and I'm translating my Yoneda to it!

Skeletons (2)
The diagrams on which I'm working can be treated as "skeletons" of categorical constructions/proofs in at least two senses.

1) The "proto-things" of the previous slides.
2) They can help us with the "the"s.

“The”

“Let’s denote by $(A\times)$ the functor that takes each object B to $A\times B$”

This means that the action of objects of $(A\times)$, $(A\times)_0$, is $B \mapsto A\times B$... $(A\times)_0 = \lambda B.(A\times B)$.
The action of morphisms of $(A\times)$, $(A\times)_1$, is not obvious.

Why do the books on CT say “$(A\times)$ is the functor that takes each object B to $A\times B$”?

Answer: because there is a way to find a natural meaning for $(A\times)_1$!
For logicians: find a proof of $(B \to C) \to (A \wedge B \to A \wedge C)$ and then apply Curry-Howard to obtain $\lambda p.(\pi p, f(\pi' p))$.
For CS’ers: find a term of type $(B \to C) \to (A\times B \to A\times C)$.

Finding a term of type such-and-such

Suppose that we know a function $f : A \to B$ and a set C. Then "f induces a function $(f{\times}C) : A{\times}C \to B{\times}C$ in a natural way".

How do we discover the function that "deserves the name" $(f{\times}C)$?

Trick: "in a natural way" usually means "using only the operations from λ-calculus", (!!!!!!!) i.e., "a λ-term".

$$\frac{f{:}A{\to}B}{(f{\times}C){:}A{\times}C{\to}B{\times}C} \quad \Rightarrow \quad \frac{A{\to}B}{A{\times}C{\to}B{\times}C} \quad \Rightarrow (...)$$

$$\dfrac{A\to B}{\overline{A\times C\to B\times C}} \quad \Rightarrow \quad \dfrac{\dfrac{\dfrac{\dfrac{[A\times C]^1}{A} \qquad A\to B}{B} \qquad \dfrac{[A\times C]^1}{C}}{B\times C}}{A\times C\to B\times C}\ 1$$

$$\Rightarrow \quad \dfrac{\dfrac{\dfrac{\dfrac{[p{:}A\times C]^1}{\pi p{:}A} \qquad f{:}A\to B}{f(\pi p){:}B} \qquad \dfrac{[p{:}A\times C]^1}{\pi' p{:}C}}{(f(\pi p),\pi' p){:}B\times C}}{(\lambda p{:}A\times C{:}(f(\pi p),\pi' p)){:}A\times C\to B\times C}\ 1$$

$$\Rightarrow \quad (f\times C) := (\lambda p{:}A\times C{:}(f(\pi p),\pi' p))$$

Internal/external, generic/particular

$$
\begin{tikzcd}
X \arrow[r, mapsto, "F_0"] \arrow[d, "g"'] \arrow[dr, phantom, "\overset{F_1}{\longmapsto}"] & FX \arrow[d, "Fg"] \\
Y \arrow[r, mapsto, "F_0"] & FY
\end{tikzcd}
\qquad
\begin{tikzcd}
B \arrow[r, mapsto, "(A\times)_0"] \arrow[d, "f"'] \arrow[dr, phantom, "\overset{(A\times)_1}{\longmapsto}"] & (A\times)B \arrow[d, "(A\times)f"] \\
C \arrow[r, mapsto, "(A\times)_0"] & (A\times)C
\end{tikzcd}
\qquad
\begin{tikzcd}
B \arrow[r, mapsto, "(A\times)_0"] \arrow[d, "f"'] \arrow[dr, phantom, "\overset{(A\times)_1}{\longmapsto}"] & A\times B \arrow[d, "\lambda p.(\pi p, f(\pi' p))"] \\
C \arrow[r, mapsto, "(A\times)_0"] & A\times C
\end{tikzcd}
$$

$$
\mathbf{A} \xrightarrow{F} \mathbf{B}
\qquad
\mathbf{Set} \xrightarrow{(A\times)} \mathbf{Set}
\qquad
\mathbf{Set} \xrightarrow{(A\times)} \mathbf{Set}
$$

$(A\times)f$ is some function with this type:
$(A\times)f : A\times B \to A\times C$.
With some practice we can find a good candidate!
$(A\times)f := \lambda p.(\pi p, f(\pi' p))$

(Not just practice! =))