# The Logic for Children Project (is trying to translate its categorical diagrams to Type Theory)

(talk @ Logic Day 2021)

By:
Eduardo Ochs →
(original author)

Selana Ochs →
(recent contributor)

Note: this presentation is a kind of
mini-rehearsal for a longer presentation titled
"Category Theory as An Excuse to Learn Type Theory"
that I submitted to the
"Encontro Brasileiro em Teoria das Categorias".
For more information on it, see:
`http://angg.twu.net/math-b.html#2021-excuse-tt`

## Logic for Children / Categories for Children

Here I will refer a lot to:

1. [PH1]: "Planar Heyting Algebras for Children" (E. Ochs, SAJL, 2019). From its abstract:

   > In a wider context these ZHAs are interesting because toposes of the form $\mathbf{Set}^{(P,A)}$ are one of the basic tools for doing "Topos Theory for Children", in the following sense. We can *define* "children" as people who think mathematically in a certain way — as *people who prefer to start from particular cases and finite examples that can be drawn explicitly, and*

> *only then generalize* — and we can define a method for working on a particular case (less abstract, “for children”) and on a general case (“for adults”) in parallel, using parallel diagrams with similar shapes; we have some ways of transfering knowledge from the general case to the particular case, *and back*. This method is sketched in the introduction.

2. [FavC]: “On my favorite conventions for drawing the missing diagrams in Category Theory”. Published on Arxiv in 2020... unpublishable? From its abstract:

> People in CT usually only share their ways of visualizing things when their diagrams cross

some threshold of mathematical relevance — and this usually happens when they prove new theorems with their diagrams, or when they can show that their diagrams can translate calculations that used to be huge into things that are much easier to visualize. The diagrammatic language that I present here lies below that threshold — and so it is a "private" diagrammatic language, that I am making public as an attempt to establish a dialogue with other people who have also created their own private diagrammatic languages.

3. [CWM]: Mac Lane's "Categories for the Working Mathematician". The standard text on CT. Very hard to read — should have 100 times more diagrams that it has, but they are left to the reader. "Normal" people start from a state in which CWM is impossible, then they switch to a state in which CWM is obvious. I got stuck studying it in many. many, many times. One of the main themes of [FavC] is formalizing "notions of obviousness", and it ends with:

   > I am especially interested in how people write when they turn their level-of-detail knob to a very high position.

4. Proof assistants based of Type Theory. From the introduction of [HOTT]:

> Type theory (...) Although it is not generally regarded as the foundation for classical mathematics, set theory being more customary, type theory still has numerous applications, especially in computer science and the theory of programming languages (...) This is the basis of the system that we consider here; it was originally intended as a rigorous framework for the formalization of constructive mathematics.

5. Haskell. From its Wikipedia page:

> At the conference on Functional Programming Languages and Computer Architecture (FPCA '87) in Portland, Oregon, there was a strong consensus that a committee be formed to define an open standard for such languages. The committee's purpose was to consolidate existing functional languages into a common one to serve as a basis for future research in functional-language design.

Haskell is based on a Type Theory that is simpler than the one in HOTT, and many universities in Europe teach

Haskell to first-year students... so there is a lot of very readable material on it.

6. Idris: essentially Haskell plus dependent types and other bells and whistles. Its type system is close enough to the one in HOTT (from my beginner's point of view). Idris can be used as a proof assistant, and the authors of [IdrisCT] have formalized some CT in Idris.

7. Discrete Mathematics. I taught DM for years, and a good part of my students entered the university without knowing how to use variables, and without knowing what is a theorem.

   Their difficulties with learning new levels of abstraction were very similar to my difficulties trying to learn Category Theory and Type Theory.

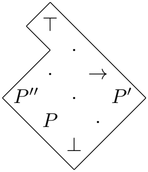
I also gave some seminar courses whose pre-requisites were only Discrete Mathematics (or not even that). I only found the right approach for writing [PH1] after these seminars — they were on $\lambda$-calculus, Heyting Algebras, S4, and Intuitionistic Logic "for children", using finite examples everywhere...

$(P \vee Q) \to (P \wedge Q)$ is not a tautology:

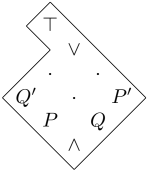
$$\underbrace{\underbrace{\underbrace{P}_{0} \vee \underbrace{Q}_{1}}_{1} \to \underbrace{\underbrace{P}_{0} \wedge \underbrace{Q}_{1}}_{0}}_{0}$$

Two classical tautologies that are not intuitionistic tautologies:

```
   32
      22
   21    12
20    11    02
   10    01
      00
```



$$\underbrace{(\underbrace{\neg\underbrace{\neg\underbrace{P}_{10}}_{02}}_{20}) \to \underbrace{P}_{10}}_{12}$$



$$\underbrace{\underbrace{\neg(\underbrace{\underbrace{P}_{10} \wedge \underbrace{Q}_{01}}_{00})}_{32} \to \underbrace{(\underbrace{\neg\underbrace{P}_{10}}_{02} \vee \underbrace{\neg\underbrace{Q}_{01}}_{20})}_{22}}_{22}$$

The connection with S4 and (order) topologies

$$\underbrace{(\underbrace{\neg\underbrace{\neg\underbrace{P}_{10}}_{02}}_{20})\to\underbrace{P}_{10}}_{12}$$

$$\underbrace{(\underbrace{\neg\underbrace{\neg\underbrace{P}_{\begin{smallmatrix} & 0 & \\ 0 & & 0 \\ 1 & & 0\end{smallmatrix}}}_{\begin{smallmatrix} & 0 & \\ 0 & & 1 \\ 0 & & 1\end{smallmatrix}}}_{\begin{smallmatrix} & 0 & \\ 1 & & 0 \\ 1 & & 0\end{smallmatrix}})\to\underbrace{P}_{\begin{smallmatrix} & 0 & \\ 0 & & 0 \\ 1 & & 0\end{smallmatrix}}}_{\begin{smallmatrix} & 0 & \\ 0 & & 1 \\ 1 & & 1\end{smallmatrix}}$$

## A trick for teaching Discrete Mathematics

$\mathbb{A}$ is our set of atoms:
the integers plus **T** and **F** (and later also ascii strings)

$\mathbb{B}$ is our set of basic mathematical objects:
$\mathbb{B}$ contains $\mathbb{A}$, and is closed by
forming finite sets and by forming finite lists
(a finite number of times)

In the first part of the course all objects that we build
are elements of $\mathbb{B}$. We use $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$ and $\mathbb{R}$ sometimes,
but expressions like $\mathbb{N} \times \mathbb{N}$ and $a \in \mathbb{N}.a^2 \geq a$ only appear
in the second part of the course.

Why?

**Layer 1: Calculating things**

...in a system with numbers, truth-values, sets and lists where everything can be calculated in a finite number of steps with almost no creativity required.

Example:

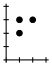
$$\begin{array}{rcl}(\forall a \in \{2,3,5\}.a^2 < 10) & = & (a^2 < 10)[a := 2] \wedge \\ & & (a^2 < 10)[a := 3] \wedge \\ & & (a^2 < 10)[a := 5] \\ & = & (2^2 < 10) \wedge (3^2 < 10) \wedge (4^2 < 10) \\ & = & (4 < 10) \wedge (9 < 10) \wedge (16 < 10) \\ & = & \mathbf{T} \wedge \mathbf{T} \wedge \mathbf{F} \\ & = & \mathbf{F}\end{array}$$

Note the substituion operator:

$(a^2 < 10)[a := 3] = (3^2 < 10)$.

**Layer 1: Set Comprehensions**
I wrote a lengthy explanation of set comprehensions, using "generators", "filters" and a "result expression". The students started by learning how to calculate things like these (note the ';' instead of a '|'; these '$\{\ldots;\ldots\}$'s are calculated from left to right!):

$$\begin{aligned} & \{\underbrace{a \in \{1,2\}}_{\text{gen}}, \underbrace{b \in \{2,3\}}_{\text{gen}}, \underbrace{a \neq b}_{\text{filt}}; \underbrace{(a,b)}_{\text{expr}}\} \\ = \ & \{(1,2),(1,3),(2,3)\} \\ = \ & \end{aligned}$$



...then $\{\, a \in \{2,3,4\} \mid a^2 < 10 \,\}$

and $\{\, 10a + b \mid a \in \{1, 2\}, b \in \{3, 4\} \,\}$.

In the rest of the talk I will use diagrams from...

[IDARCT]:
`http://angg.twu.net/math-b.html#idarct`
`http://angg.twu.net/LATEX/idarct-preprint.pdf`

[PH1]:
`http://angg.twu.net/math-b.html#zhas-for-children-2`
`http://angg.twu.net/LATEX/2017planar-has-1.pdf`

[FavC]:
`http://angg.twu.net/math-b.html#favorite-conventions`
`http://angg.twu.net/LATEX/2020favorite-conventions.pdf`

[HOTT] [Zav20] [KLN04] [Som00]

## References

[CWM] S. Mac Lane. *Categories for the Working Mathematician (2nd ed.)* Springer, 1997.

[FavC] E. Ochs. "On my favorite conventions for drawing the missing diagrams in Category Theory". `http://angg.twu.net/math-b.html#favorite-conventions`. 2020.

[HOTT] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. `http://saunders.phil.cmu.edu/book/`

hott-online.pdf. Institute for Advanced Study, 2013.

[IDARCT] E. Ochs. “Internal Diagrams and Archetypal Reasoning in Category Theory”. In: *Logica Universalis* 7.3 (Sept. 2013). http://angg.twu.net/math-b.html#idarct, pp. 291–321.

[IdrisCT] F. Genovese et al. *idris-ct: A Library to do Category Theory in Idris*. 2019. arXiv: 1912.06191 [cs.LO].

[KLN04] F. Kamareddine, T. Laan, and R. Nederperlt. *A Modern Perspective on Type Theory*. Kluwer, 2004.

[PH1] E. Ochs. “Planar Heyting Algebras for Children”. In: *South American Journal of Logic* 5.1 (2019).

http://angg.twu.net/math-b.html#zhas-for-children-2, pp. 125–164.

[Som00] G. Sommaruga. *History and Philosophy of Constructive Type Theory*. Springer, 2000.

[Zav20] V. Zavialov. "Haskell to Core: Understanding Haskell Features Through Their Desugaring". https://youtu.be/fty9QL4aSRc. 2020.