Category Theory of Logic

by Sven Nilsen, 2019

There are many languages of logic, e.g. first order logic, linear logic, many-value logic and probabilistic logic. Is it possible to talk about all families of logical languages in general? In this paper I formalize the general family of logic in Category Theory, by starting with the definition of a category, adding two simple axioms and then I show that this becomes a groupoid containing two objects "true" and "false". The interpretation of any logical language extends this special groupoid such that for every extension, there exists an inverse morphism leading back to "true" and "false".

I had a strange dream that I was in a room full of mathematicians. There was a discussion about mathematics and the overall consensus was that mathematics is about theorem proving and building incrementally on other people's knowledge. All of a sudden I rose up, protested, claiming that mathematics has little to do with such activities, those who wanted to do "real mathematics" were welcome, then I walked up to the front of the room to teach them a new way to think about logic.

Yeah, it was a strange dream and this would be very embarrassing in real life. :P

In the dream I made a picture with chalk on a green blackboard, claiming that all of logic could be derived from a single arrow: $\mathbf{0} \to \mathbf{1}$. I used 3D axes as an example. The rest is Category Theory, but you can not prove it is true, since this description of logic follows from deep intuitive insight, not by taking incremental steps. Mathematics is about finding such clear but perhaps unexplainable definitions.

It was so real and convincing to myself and the mathematicians in the room that when I woke up, I thought about it for a while and then picked up a pen and paper to test it. Could the dream-version of myself be right? Turns out: Almost. The dream-me did not see the requirement of cyclic composition.

The general family of logic is a category with the following additional axioms:

$\forall$ a, b, c { $\exists a \to b \wedge \exists b \to c \Rightarrow \exists c \to a$ }	Cyclic composition of morphisms
$\exists \mathbf{0} \to \mathbf{1}$	Morphism from "false" to "true"

The notation `$\exists a \to b$` means "there exists a morphism from `a` to `b`".

`**0**` is the "false" element and `**1**` is the "true" element of a logical language.

The axioms of a category can be written as following (composition has a weaker form here):

$\forall$ a, b, c { $\exists a \to b \wedge \exists b \to c \Rightarrow \exists a \to c$ }	Composition of morphisms
$\forall$ a { $\exists a \to a$ }	Identity morphisms

Now, this might not look like logic at all. However, think about it like this: A category might contain any number of objects. A morphism between two objects corresponds to something in the language that makes sense. Rules for morphisms means talking about how things make sense in logic.

Using cyclic composition of morphisms, one can prove that for every morphism, there exists a morphism heading in the opposite direction:

$\forall$ a, b, c { $\exists a \to b \wedge \exists b \to c \Rightarrow \exists c \to a$ }	b = c
$\forall$ a, b { $\exists a \to b \wedge \exists b \to b \Rightarrow \exists b \to a$ }	Notice identity morphism `$\exists b \to b$`
$\forall$ a, b { $\exists a \to b \Rightarrow \exists b \to a$ }	Holds for any category

With other words, the category of logic is its own co-category.
The technical term for this is that it is a *groupoid*.

From this, it follows that:

$\exists \mathbf{0} \to \mathbf{1}$ => $\exists \mathbf{1} \to \mathbf{0}$

Think about this as: For any logical language, there is some way to turn “false” into “true” and “true” into “false”. Like a switch. Or, the function `not` in Boolean algebra. Or, `1 – p` in probability theory.

Let’s say we create a logical language where one can construct tuples of `0` and `1`:

(0, 0)
(0, 1)
(1, 0)
(1, 1)

In Slot Lambda Calculus, one can think of these as functions:

(_, _)(0)(0)	=	(0, _)(0)	=	(0, 0)
(_, _)(0)(1)	=	(0, _)(1)	=	(0, 1)
(_, _)(1)(0)	=	(1, _)(0)	=	(1, 0)
(_, _)(1)(1)	=	(1, _)(1)	=	(1, 1)

By using functions from Slot Lambda Calculus as morphisms, we have:

$\exists 0 \to$ (0, _)	(_, _)(0)
$\exists 1 \to$ (1, _)	(_, _)(1)
$\exists 0 \to$ (0, 0)	(0, _)(0)
$\exists 1 \to$ (0, 1)	(0, _)(1)
$\exists 0 \to$ (1, 0)	(1, _)(0)
$\exists 1 \to$ (1, 1)	(1, _)(1)

Since this is a logical language, there must exist inverses, which naturally are `fst` and `snd`:

$\exists$(0, _) $\to$ 0	fst((0, _)) = 0
$\exists$(1, _) $\to$ 1	fst((1, _)) = 1
$\exists$(0, 0) $\to$ 0	snd((0, 0)) = 0
$\exists$(0, 1) $\to$ 1	snd((0, 1)) = 1
$\exists$(1, 0) $\to$ 0	snd((1, 0)) = 0
$\exists$(1, 1) $\to$ 1	snd((1, 1)) = 1