# Category Theory in Free Logic

Lucca Tiemens

July 21, 2020

In this document I want to introduce the notions of a functor, a natural transformation, a cartesian category and a cartesian closed category in free logic. All of these are already implemented in Isabelle/HOL, though I have not proven facts about them. This summarizes my work done on the project during the last weeks. Also, I have almost finished the implementation of the category of sets as I believe this category to be important to have later on.

Note that I will denote quantifiers that range only over existing elements by $\forall_f$ and $\exists_f$.

**Definition 1** (Type)**.** A morphism $m$ is called a type if $\mathrm{dom}(m) \cong m$.

**Definition 2** (Functor)**.** Given two categories $\mathbf{A}$ and $\mathbf{B}$, a functor between $\mathbf{A}$ and $\mathbf{B}$, denoted by $F : \mathbf{A} \to \mathbf{B}$, is such that

1. $\forall x \in \mathbf{A} : Ex \iff E(Fx)$
2. $\forall x \in \mathbf{A} : \text{type } x \implies \text{type } (Fx)$
3. $\forall x \in \mathbf{A} : F(\mathrm{dom}(x)) \cong \mathrm{dom}(Fx)$
4. $\forall x, y \in \mathbf{A} : E(x \cdot y) \implies (F(x \cdot y) \cong (Fx) \cdot (Fy))$

**Definition 3** (Natural Transformation)**.** A natural transformation between two functors $F : \mathbf{A} \to \mathbf{B}$ and $G : \mathbf{A} \to \mathbf{B}$ consists of a map $\upsilon : \mathbf{A} \to \mathbf{B}$ such that

$$\forall_f x \in \mathbf{A} : \upsilon(\mathrm{cod}(x)) \cdot (Fx) \simeq (Gx) \cdot (\upsilon(\mathrm{dom}(x)))$$

**Definition 4** (Category with binary products)**.** A category with binary products $\mathbf{C}$ is a category together with a map $\times : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$ satisfying

1. $\forall_f x, y \in \mathbf{C}\, \exists_f p1, p2 \in \mathbf{C} : (\text{product } x\, y\, (x \times y)\, p1\, p2)$
2. $E(x \times y) \implies (Ex \wedge Ey)$

As you suggested, Dana, in this way the product is canonical by the map $\times$ that we specify.

I adjusted the definition of a product a little bit because the definition of an exponential requires that we have the projection maps at hand. Therefore, the definition of a product now takes the two projection maps as the last two parameters.

I have not introduced the maps fst and snd yet as I was not sure if it will be enough to specify their behavior by demanding fst $(a \times b) \cong a$ and snd $(a \times b) \cong b$ because, if not applied to a product, fst and snd will return an arbitrary value.

**Definition 5** (Equalizer)**.** An equalizer in a category $\mathbf{C}$ between two parallel morphisms $f : A \to B$ and $g : A \to B$ is a morphism $e$ such that

1. $f \cdot e \simeq g \cdot e$
2. $\forall_f z \in \mathbf{C} : f \cdot z \simeq g \cdot z \implies \exists_f ! u : A \to B \in \mathbf{C} : e \cdot u \simeq z$

Question: Do you think it will be useful to introduce a function symbol for the equalizer?

**Definition 6** (Cartesian Category)**.** A Cartesian category $\mathbf{C}$ is a category with binary products together with a type $\mathbf{1}$ such that

1. $\forall f : A \to B, g : A \to B \in \mathbf{C} : \exists e \in \mathbf{C} : (\text{equalizer } f\, g\, e)$
2. final $\mathbf{1}$

**Definition 7** (Exponential)**.** In a category with binary products we define

$$\begin{aligned}
&\text{exponential } a\, b\, c\, \epsilon :\iff \exists_f p1, p2 \in \mathbf{C} : \\
&\qquad \text{product } (a\, c\, \text{dom}(\epsilon)\, p1\, p2) \wedge \epsilon : (a \times c) \to b \wedge \\
&(\forall_f z, f \in \mathbf{C} : \exists_f q1, q2 \in \mathbf{C} : \text{product } (z\, a\, \text{dom}(f)\, q1\, q2) \wedge f : (z \times a) \to b \wedge \\
&\qquad (\exists_f ! \hat{f} : z \to c \in \mathbf{C} : (\exists_f ! \mathtt{j} : \text{dom}(f) \to \text{dom}(\epsilon) : \\
&\qquad\qquad p2 \cdot \mathtt{j} \simeq \hat{f} \cdot q1 \wedge p1 \cdot \mathtt{j} \simeq q2 \wedge \epsilon \cdot \mathtt{j} \simeq f)))
\end{aligned}$$

I am convinced that the above definition is correct because I first derived it by my own, studying the usual definition of an exponential in diagram form and then I also found the above definition in Freyd and Scedrov's book *Categories, Allegories* (tough of cause in their formalization but it is usable for us - so are all their definitions I have looked at so far by the way).

By integrating the product map $\times$ into the definition of an exponential, I think we achieve a good canonical interplay between exponentials and products - thus we do not need to reason up to isomorphism so far. This should enable the definition of the product functor and the proof that exponentiation is functorial (I have not tried that yet).

All of this allows the definition of a Cartesian Closed Category:

**Definition 8** (Cartesian Closed Category)**.** A *CCC* category $\mathbf{C}$ is a category with binary products and a map $exp : \mathbf{C} \times \mathbf{C} \to \mathbf{C}$ such that

1. $\forall_f a, b \in \mathbf{C} : \exists_f \epsilon :$ exponential $(a\, b\, (\exp\ a\, b)\, \epsilon)$
2. $E(\exp\ a\, b) \implies E\, a \wedge E\, b$¸

A Cartesian closed category as defined above is a Cartesian category. This is not proven yet in Isabelle/HOL. If we want to reason more about *CCC*'s it will probably be easier to introduce a second definition which has the final object in the signature.

**General comments:**

The reason why I use Typeclasses in Isabelle/HOL at the moment is the fact that it makes writing down new definitions easier. In fact, I believe a Typeclass does not add any more expressive power but it is simply convenient to use. The advantage is that the presentation of the theory will later on be more readable and easier to follow. Furthermore, current Isabelle/HOL theories all try to use Typeclasses as far as I have experienced. So it is also *modern* style in a sense. Of cause sledgehammer and ATP's might not work as good anymore. I believe the best way to handle this is to have a version which uses locales and a version that uses classes (the proofs should be very similar for both cases). Same goes for abbreviations versus definitions. In order to study how far ATP's can go, the locale version should be used and for presentation the class version should be used. Then one can also compare what the differences are.

To give you an example for what I mean by more readable (and therefore also easier to use for exploration for new definitions) consider natural transformations between two functors.

**Example 1.** If we use Typeclasses we can write down

$$\text{naturalTransformation}\ \mathcal{F}\, \mathcal{G}\, \upsilon.$$

The additional information that for example the domain of $\mathcal{F}$ is a category is stored in a Typeclass variable found in the definition of isFunctor .

If we use Locales, the definition of a natural transformation explicitly needs the categories on which the functors are defined. So if we want to assume to have a natural transformation between $\mathcal{F}$ and $\mathcal{G}$, we need to write down

$$naturalTransformation\, \mathcal{F}\, \mathcal{G}\, \upsilon\, dom_1\, cod_1\ \cdot_1\ dom_2\, cod_2\ \cdot_2 .$$

Do you have more thoughts on this, Christoph?

The reason for using ZFC is quite simple: I want to have the category of sets. I do not know of another way to accomplish that without axiomatizing ZF for a new type.