The Category of Categories is Cartesian Closed

Maarten M Fokkinga

CWI, Amsterdam, and University of Twente, Enschede

Version of Feb 5, 1991

The category of categories is cartesian closed. This means amongst others that \( \text{curry}(\ddagger) \) is well defined for any bi-functor \( \ddagger \), having the properties that we expect it to have. The \textit{sectioning} notation \( x\ddagger \) may be used to denote “\( x \) subject to \( \text{curry}(\ddagger) \)” (for object or morphism \( x \)); it follows that \( f\ddagger \) is a natural transformation from \( A\ddagger \) to \( B\ddagger \), whenever \( f : A \to B \).

Introduction

The category of categories, \( \mathcal{C} \), has all (small?) categories as objects and all functors as morphisms. Cartesian closedness of \( \mathcal{C} \) means

- there exists a category \( 1 \) that is \textit{final} in \( \mathcal{C} \),
- for any two categories \( A \) and \( B \) there exists a category \( A \times B \) and suitable projection functors that together constitute a \textit{product} in \( \mathcal{C} \), and
- for any two categories \( A \) and \( B \) there exists a category \( A \to B \) and functor \( @_{A,B} : A \times (A\to B) \to B \) and, for any functor \( \ddagger : A \times B \to C \), a functor \( \ddagger^\ddagger : A \to (B\to C) \) that together constitute an \textit{exponent} in \( \mathcal{C} \).

Once the formal requirements are laid down, most of the definitions are straightforward and present no surprises if the point-wise construction of the final object, the products, and the exponents within \textbf{Set} are known. Also, the verification that the required typing and equations are fulfilled is a matter of routine. The “only” difference with \textbf{Set} is this: in \textbf{Set} a morphism is just a single function, whereas in \( \mathcal{C} \) a morphism is a functor and therefore both a function from objects to objects and a function from morphisms to morphisms.

Let us consider the construction for exponents in some more detail. We shall use the following notation and naming convention, unless stated explicitly otherwise.

- \( A, B, \ldots \) vary over categories (i.e. \( A \in \text{Obj}(\mathcal{C}) \));
- \( F, G, \ldots \) vary over functors, typically \( F : A \to B \) (i.e., \( F \in \text{Hom}_\mathcal{C}(A, B) \));
- \( x.F \) denotes “\( x \) subject to \( F \)”, and \( x.(F \cdot G) = (x.F).G \);
\[ \vdash : A \times B \to C \text{ in } C \text{ and } x \vdash y = (x, y). \]

\( A, A', \ldots \) vary over objects of \( A \) (i.e., \( A \in \text{Obj}(A) \)) and so on;

\( f, g, \ldots \) vary over morphisms, typically \( f : A \to A' \) in \( A \) (i.e., \( f \in \text{Hom}_A(A, A') \)) and \( g : B \to B' \) in \( B \);

composition of morphisms in \( A \) and so on is denoted \( f ; f' \).

**Exponents**

**Exponent, currying** Given categories \( A \) and \( B \) we define the category \( A \to B \) to be the well-known category of functors from \( A \) to \( B \) whose morphisms are natural transformations. Given a (bi-) functor \( \vdash : A \times B \to C \) we define the (mono-)functor \( \vdash : A \to (B \to C) \) as follows.

\[
A.\vdash = \text{the functor from } B \to C \text{ given by}
\]

\[ B.(A.\vdash) = A \vdash B \]

\[ g.(A.\vdash) = \text{id}_A \vdash g : B.(A.\vdash) \to B'.(A.\vdash) \]

\[
f.\vdash = \text{the natural transformation from } A.\vdash \text{ to } A'.\vdash \text{ given by}
\]

\[ B.(f.\vdash) = f \vdash \text{id}_B : A \vdash B \to A' \vdash B \text{ in } C : \]

The requirements for \( A.\vdash \) to be a functor, and for \( f.\vdash \) to be a natural transformation, are easily verified. We can extend the above definition of \( f.\vdash \) (as a mapping from objects to morphisms) with a mapping from morphisms to morphisms as follows. (Here \( g \cdot \varphi \) denotes “\( g \) subject to \( \varphi \).”)

\[
g \cdot (f.\vdash) = f \vdash g \]

\[ = (\text{id}_A \vdash g) ; (f \vdash \text{id}_{B'}) \]

\[ = g.(A.\vdash) ; B'.(f.\vdash) \]

\[ = B.(f.\vdash) ; g.(A'.\vdash) : \]

This is no surprise since we can do so in general for any natural transformation \( \varphi : F \to G \) in \( B \to C \) (with \( F, G : B \to C \) in \( C \)):

\[
g.F ; \varphi_{B'} = \varphi_B ; g.G =: g \cdot \varphi \]

for any \( g : B \to B' \) in \( B \).

**Sectioning** We may use the notation \( x.\vdash \) for \( x.\vdash \). It has been defined above for both objects \( x \) and morphisms \( x \), and we have seen that \( A.\vdash \) is a functor and \( f.\vdash \) is a natural transformation.

When object \( A \) in \( A \) is also used to denote the identity morphism \( \text{id}_A : A \to A \) and the constant functor \( A^\bullet : X \to A \) (mapping an object to \( A \) and a morphism to \( \text{id}_A \)), then we can summarize all four definitions of \( \vdash \) by

\[
y.(x.\vdash) = x \vdash y \text{ in } C \]

for any object and morphism \( x \) in \( A \) and any object and morphism \( y \) in \( B \). (Notice that there is a syntactic ambiguity in \( f ; A.F \) and \( (A.F) ; F' \) but no semantic ambiguity, since \( \text{id}_A.F = \text{id}_{A,F} \).)
Evaluation  We also need to define for any two objects \( A \) and \( B \) in \( C \) an evaluation functor \( @_{A,B} : A \times (A \rightarrow B) \rightarrow B \). As a mapping on objects its definition suggests itself; as a mapping on morphisms it might be a very little bit surprising.

\[
\begin{align*}
(A,F).@ &= A.F \quad \text{in } B \\
(f,\varphi).@ &= f \cdot \varphi : A.F \rightarrow A'.F \quad (= f.F; \varphi = \varphi; f.G);
\end{align*}
\]

for \( f : A \rightarrow A' \) in \( A \) and \( \varphi : F \rightarrow G \) in \( A \rightarrow B \). In order to fully complete the proof that these constructions do constitute an exponent, the following equivalence has to be satisfied:

\[
\begin{align*}
F = \uparrow &\equiv F \times I_{B \rightarrow C} : @_{B,C} = \uparrow
\end{align*}
\]

for all \( F : A \rightarrow B \) in \( C \). Since \( @ \) is defined pointwise one can easily check the equivalence by extensionality.

Acknowledgement  I have had an instructive discussion with Lambert Meertens and Jaap van der Woude on this topic.