Applied Category of Sets

by Sven Nilsen, 2022

In this paper I introduce 4 thumb rules that might help the formal understanding of the Applied Category of Sets, as distinct from the abstract mathematical object that is Category of Sets.

The Category of Sets[1] is of vital importance in mathematics, yet people who study math might often treat it superficially and focus only on properties that are relevant for theorem proving. I would like to broaden the scope in this context such that the Category of Sets can be studied as a tool, not just as an abstract mathematical object. I name this category “Applied Category of Sets” to distinguish it from the usual abstract mathematical objects which is Category of Sets.

The following four thumb rules hold for the Applied Category of Sets:

1. Only morphisms are named at first, objects are named as an afterthought
2. Every morphism has an explicit operator
3. All arguments have constant semantics
4. Composition yields externalized partial evaluation of constants

These thumb rules are not provable from the axioms of Category Theory plus the knowledge that the objects in the Category of Sets have sets as objects and functions as morphisms. Therefore, this additional knowledge is important to reason about how Category of Sets is applied in practice.

1. Only morphisms are named at first, objects are named as an afterthought

In the axioms of Category Theory, there is no nuance in the mathematical language used to define these axioms that can express the nuances of importance toward objects or toward morphisms. The language is designed in a such way that objects might seem just as important as morphisms and additional information is needed to tell there is a nuance there.

In the Applied Category of Sets, the names of morphisms are very important and the names of objects are not important. For many applications, it is possible to ignore the type of objects and only reason about e.g. algebraic operators abstractly, without knowing the concrete type they operate on.

2. Every morphism has an explicit operator

This can seem obvious at first, but counter-intuitively, there is no reason that the name given to some morphism should be associated with the operator that the morphism represents. It is possible to give a morphism a random name.

For example:

$$\text{add} : T \times T \to T$$

Here, the `add` operator takes the Cartesian product[2] of two `T`s and produces a `T`. While `T` is strictly a type, it is usually thought of as a set and this is how Category of Sets is used to reason about functions.

Since `add` is identified by the morphism using this name and its surrounding morphisms, it can in principle be given an entirely different name e.g. `blarb` without affecting reasoning that is sufficient for theorem proving. This is like renaming a proposition in a proof: The proof holds, no matter what names are used. There is only one place where the name `add` is used, so its name is not important. Or, to be more precise, the local relevance of the name equals the global relevance of the name.

However, in applied mathematics, one studies the Category of Sets in a such way that composition reveal insights about equivalence relationships between morphisms, or about optimization. In order to do this, one argument on binary operators needs to be lifted to a constant argument, which kind of extends the name of morphisms. This means that a single category might contain multiple operators used in multiple places, such that the name of the operator identifies these multiple uses of operators with each other.

For example, the algebraic law of distribution (the bold `**x**` here means a variable/object):

$$(\mathbf{x} + a) \times b = \mathbf{x} \times b + a \times b$$

Can be expressed as the following:

$$(\times b) \cdot (+ a) <=> (+ (a \times b)) \cdot (\times b)$$

Since the arguments `a` and `a × b` differ to `+`, one needs a symbol to express the operator in order to identify these two instances of `+`. Where it before was possible to put `+` in a single place, making the local relevance of the name equal to the global relevance, it is now needed to use a symbol consistently due to the asymmetry between local and global relevance.

This structure is added on to the Category of Sets, making Applied Category of Sets different, in the way that explicit operators are used per morphism.

3. All arguments have constant semantics

All arguments in Applied Category of Sets has semantics that is equal to that of constants. This means, when writing e.g. `(+ 2)`, one can treat the argument as extending the name of the morphism by using the explicit operator `+` together with `2`. This holds for any number of arguments and abstractions such as `f(a, b) = (× b) · (+ a)`. With other words, arguments are used to identify the morphism.

4. Composition yields externalized partial evaluation of constants

In the Applied Category of Sets, there is partial evaluation which is proof irrelevant, except that it computes on constant inputs and evaluates to some constant. This proof irrelevance of the particular form of partial evaluation is a kind of “externalization” of some part of the composition.

This might seem hard to understand at first, but it is important to remember that when binary operators lift one argument, there are also other parts that get lifted, corresponding to evaluations.

For example:

$$(+\ b) \cdot (+\ a) <=> (+\ (a + b))$$

Here, the operator `a + b` lives “outside” the scope of the category structure. One can think about this as being able to use an external function call to compute `a + b`.

Another example, based on the law of distribution:

$$(\times\ b) \cdot (+\ a) <=> (+\ (a \times b)) \cdot (\times\ b)$$

Using abstraction:

$$\therefore \quad f(a, b) <=> g(a \times b, b)$$

$$\because \quad f(x, y) = (\times\ y) \cdot (+\ x)$$
$$\because \quad g(x, y) = (+\ y) \cdot (\times\ x)$$

The partial evaluation results in constants which are used to identify the morphisms, together with the explicit operators. It does not matter which computation is performed, only that the computation has this nature of constant semantics. Therefore, one can view this partial evaluation as “externalized” from the category structure.

The evaluation can be thought of as being part of the categorical structure originally, but when it gets lifted the particular computational properties become proof irrelevant, except the ability to evaluate constant outputs from constant inputs. This is what is meant by “externalization”.

References:

[1] “Category of sets”
Wikipedia
https://en.wikipedia.org/wiki/Category_of_sets

[2] “Cartesian product”
Wikipedia
https://en.wikipedia.org/wiki/Cartesian_product