Category Theory Presentation

Michael Walker

13th February, 2013

Contents

List of Figures

1 Categories

A category is a collection of objects and arrows, where each arrow has a source object and a target object (sometimes called the domain and codomain), where every object A has an identity arrow $id_A : A \to A$ and, for every pair of arrows f and g with $\operatorname{cod} f = \operatorname{dom} g$, the composite $g \circ f$ is defined, where composition is associative.

The simplest example of a category is a discrete category. This is a category with no non-identity arrows, and so it is completely defined by its object collection. We can immediately turn any set into a category by considering it as the object collection of a discrete category.

$$\begin{tikzcd}
a \arrow[loop above] & b \arrow[loop above] & c \arrow[loop above]
\end{tikzcd}$$

Figure 1: Discrete category for the set $\{a, b, c\}$.

For a slightly more complex example, consider the category **Set**, the category of all sets with arrows being functions between them. We can illustrate the composition of arrows with a very small subcategory of **Set** as follows. The identity arrows have been omitted for brevity.

$$\begin{tikzcd}
\{1,2,3\} \arrow[r, "(\times2)"] \arrow[dr, "(+1)\circ(\times2)"'] & \{2,4,6\} \arrow[d, "(+1)"] \\
 & \{3,5,7\}
\end{tikzcd}$$

Figure 2: Composition of arrows in **Set**

For every category C we can form its opposite category (sometimes called dual category), denoted by C_{op}, which is formed by reversing all of the arrows in C. That is, for every arrow $f : A \to B$ in C, there is an arrow $g : B \to A$ in C_{op}. To continue the example from above, the opposite of that category would be

$$\begin{tikzcd}
\{1,2,3\} & \{2,4,6\} \arrow[l, "(\div2)"] \\
 & \{3,5,7\} \arrow[u, "(-1)"] \arrow[ul, "(\div2)\circ(-1)"]
\end{tikzcd}$$

Figure 3: Composition of arrows in $\mathbf{Set}_{op}$

As would be expected, all of the results for normal categories apply to opposite categories, and vice versa.

We can actually work with categories that just have arrows and no objects, where arrows have other arrows as their sources and targets. We construct this by, for every arrow f, having arrows u and u' such that $f \circ u = f$ and $u' \circ f = f$. These arrows are equivalent to the identity arrows of what would be the corresponding objects in a normal category. However, working with categories in this way is usually just a needless confusion as the two theories are equivalent.

2 Natural Transformations

Natural transformations are often called morphisms on functors. Given two functors S and T between the same categories, a natural transformation τ assigns to each object in the category an arrow in such a way that the following figure commutes

$$\begin{tikzcd}
Sa \arrow[r, "\tau a"] \arrow[d, "S(f)"] & Ta \arrow[d, "T(f)"] \\
Sb \arrow[r, "\tau b"] & Tb
\end{tikzcd}$$

Figure 4: Commutative diagram for natural transformations

We call τa, τb, ... the components of the natural transformation. When each component is invertible, we call the natural transformation a natural isomorphism, or a natural equivalence. This lets us immediately define a notion of equality for categories. Two categories A and B are equal when there exist functors $S : A \to B$, $T : B \to A$, and natural isomorphisms $I_A \cong T \circ S$ and $I_B \cong S \circ T$, where I_A and I_B are the identity functors for categories A and B.

As an example of natural equivalences, consider the following: every category is equivalent to its opposite category. That is, we have two functors $OP : C \to C_{op}$, $OP' : C_{op} \to C$, and can find natural isomorphisms $I_C \cong OP' \circ OP$ and $I_{C_{op}} \cong OP \circ OP'$. In this case, the isomorphisms are trivial to find, and in fact the isomorphisms $\tau : I_C \to OP' \circ OP$, $\tau c = c$ and $\theta : I_{C_{op}} \to OP \circ OP'$, $\theta c = c$ suffice.

$$\begin{tikzcd}
I_C\ a \arrow[r, leftrightarrow, "\tau a"] \arrow[d, "I_C(f)"] & OP' \circ OP\ a \arrow[d, "OP' \circ OP(f)"] & I_{C_{op}}\ a \arrow[r, leftrightarrow, "\theta a"] \arrow[d, "I_{C_{op}}(f)"] & OP \circ OP'\ a \arrow[d, "OP \circ OP'(f)"] \\
I_C\ b \arrow[r, leftrightarrow, "\tau b"] & OP' \circ OP\ b & I_{C_{op}}\ b \arrow[r, leftrightarrow, "\theta b"] & OP \circ OP'\ b
\end{tikzcd}$$

Figure 5: Every category is equivalent to its opposite category

Using natural transformations, we can form what is called a functor category. Given two categories C and D, we denote by D^C the category with objects all covariant functors from C to D and with arrows the natural transformations between these functors. These are useful because some common categories are actually just functor categories in disguise.

As an example of functor categories, consider the category of all directed graphs. A directed graph consists of a set of nodes and a set of edges, with two functions from the node set to the edge set. Thus, the category of all directed graphs is simply the category $\mathbf{Set}^C$, where C is the category with two objects and two arrows from one of the objects to the other.

Now we shall look briefly at how category theory can be used in the fields of maths, computer science, music, quantum physics, and biology.

3 Applications

3.1 Computer Science

In computer science, we can consider certain parts of programming languages in the light of category theory. Given a functional programming language, we can form the category which has as objects the types of the language and as arrows the functions between types. Defining identity functions and function composition is trivial, and so this does form a category.

Thus, any results that can be proved for categories in general apply to the specific case of programming language types and, furthermore, by incorporating more concepts from category theory we can enrich our languages and make some tasks much simpler.

For example, we can incorporate functors into our languages to denote values which have some extra context associated with them. In the Haskell programming language, there is a functor called *Maybe* which is used to indicate computations which may fail. Lists, trees, and other data structures can all be considered as functors as well in some cases, and suddenly our language is far more powerful as we can apply, say, a function on numbers to a tree of numbers, via the arrow function of the functor. Without functors, we would have to write a separate function for each data structure, even if they all did the same basic operation.

3.2 Physics

Quantum systems can be considered as categories. Objects are systems and arrows are operations which change the system. Clearly, we can perform one operation and then perform another, and composing operations is associative. Furthermore, if we add the operation of doing absolutely nothing, we get the identity arrows. And thus, we form a category.

Specifically, we use a symmetric monoidal category, which is a category equipped with a symmetric monoidal tensor: an operation that effectively lets us consider the parallel composition of two systems, in addition to just their sequential composition. This is the tensor product of quantum mechanics. It turns out that by expressing the requirements of bifunctionality, symmetry, associativity, and the left and right units of this operation categorically, we can deduce nontrivial results about how operations and systems may be composed.

4 Conclusion

Category theory is called "abstract nonsense" for a reason. It looks at all of the current mathematical abstractions, and takes them one step further and, in doing so, yields a rich and powerful theory with which we can unify different concepts and find applications in even very practical areas.

We have only given a brief introduction to category theory today, but hope that you now have some appreciation for how it can be used in many different fields. It is only a very simple theory, with few axioms, but because of its simplicity manages to be incredibly general and have great unifying power over mathematics. Once, set theory was this unifying structure, but it didn't prove general enough. Perhaps, now, category theory will succeed where it failed in providing a universal groundwork.

Thank you.