Capstone Summary

Daniel Sinderson

March 2024

What I'm Doing and Why

For my senior capstone project in mathematics I'm learning category theory, specifically with an interest in recent work applying category theory to dynamical systems.

Category theory studies composition. By studying and abstracting notions of composition, categories manage to encapsulate a shockingly flexible and wide-reaching language for describing and working with structure of all sorts. Much like the study of sets and the set membership of elements turns out to be capable of formalizing all of mathematics, so too does category theory, except from a bird's eye point of view: the details of a particular field go out of focus and only its high level structural patterns remain. This is useful. High levels of abstraction provide high levels of generality, and a general, common language of how things are structured is useful for both organizing thought and sharing it. This is the primary reason I'm drawn to category theory.

The second reason is its potential for opening a broader class of thinking and phenomena to mathematical rigor. I have a Bachelor' degree in anthropology. I still read 5-10 ethnographies and works on social theory every year, and I'm consistently impressed by the qualitative depth and intellectual creativity of the field. I'm also consistently left wanting more by the body of theory underpinning the research and how it's interpreted. Not because it's not brilliant, but because of the imprecision of its language, and the opportunities that that affords for misunderstanding, misinterpretation, and misuse. Part of my wandering into category theory then and its applications to systems theory is a search for a better language of structure: a language that's flexible and abstract enough to hold the complications and self-referentiality of human relations, and rigorous and precise enough to do so clearly. I've only just arrived, but the view is promising and, more importantly, its spectacular.

Category Theory

The first goal of my project was to get a working understanding of category theory up to and including the Yoneda Lemma. I split this material up into four section: the basics (categories, functors, and natural transformations), universal properties and limits, structure between categories (subcategories, representability, equivalence, natural isomorphism, adjunctions, and monads), and monoidal categories. As representative samples here are my definitions for a category and a monoidal category.

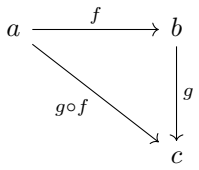
Definition 1 (Category)**.** A category $\mathcal{C}$ is defined by the following:

1. $\mathcal{C}$ contains a collection of objects $ob(\mathcal{C})$. We'll denote that an object is in a category using set notation: $c \in \mathcal{C}$.

2. For any two objects a, $b \in \mathcal{C}$ there is a collection of morphisms, or arrows, between those objects $\mathcal{C}(a, b)$ called the homset. This is short for "set of homomorphism." We'll denote an element $f \in \mathcal{C}(a, b)$ using function notation: $f : a \to b$.

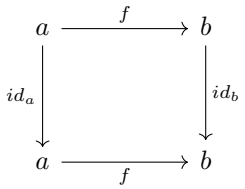
3. Every object $a \in \mathcal{C}$ has a morphism to itself $id_a : a \to a$ called its identity. This morphism doesn't do anything. It's like multiplying a number by 1.

4. For every two morphisms $f : a \to b$ and $g : b \to c$ there's a third morphism $g \circ f : a \to c$ that's their composition. The circle is the symbol for function composition and $g \circ f$ is read "g after f."

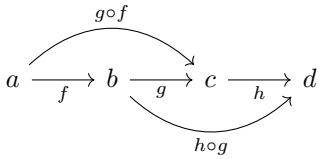


These objects and morphisms are then under the following two constraints:

1. (Unitality) Any morphism $f : a \to b$ can be composed with the identity morphisms of a and b such that $f \circ id_a = id_b \circ f = f$. This makes the following diagram commute. What this means is that both paths from a to b here are equivalent.



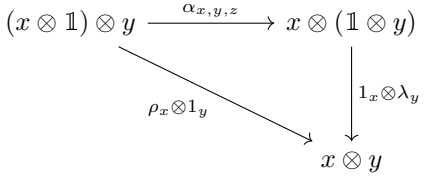
2. (Associativity) For any morphisms $f : a \to b$, $g : b \to c$, and $h : c \to d$, $h \circ (g \circ f) = (h \circ g) \circ f$. Since it doesn't matter what order we apply the morphisms, we write this $h \circ g \circ f$.

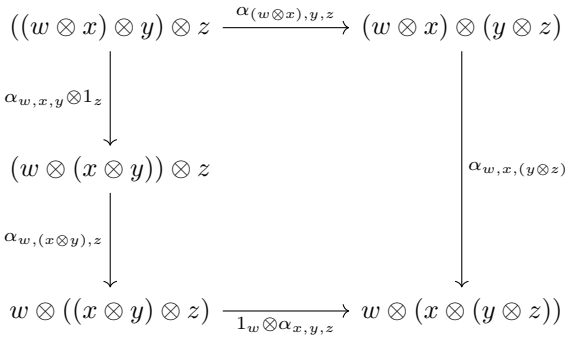


Definition 2 (Monoidal Categories)**.** A category $\mathcal{C}$ is monoidal if the following exist.

1. A functor $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ called the monoidal product.
2. An object $\mathbb{1} \in ob(\mathcal{C})$ called the unit.
3. A natural isomorphism $\alpha : (a \otimes b) \otimes c \to a \otimes (b \otimes c)$ called the associator, with components $\alpha_{x,y,z} : (x \otimes y) \otimes z \to x \otimes (y \otimes z)$.
4. A natural isomorphism $\lambda : \mathbb{1} \otimes a \to a$ called the left unitor with components $\lambda_x : \mathbb{1} \otimes x \to x$.
5. A natural isomorphism $\rho : a \otimes \mathbb{1} \to a$ called the right unitor with components $\lambda_x : x \otimes \mathbb{1} \to x$.

All of the above must exist such that the following two diagrams, called the triangle identity and the pentagon identity, commute.





Categorical System Theory

For the second part of my project I wanted to dig in to some of the recent work on applying category theory to fields outside of mathematics. One such research program is categorical systems theory, which in turn seems to set the groundwork and vocabulary for attempts to categorify game theory and cybernetics. Since all of these are right in line with my interest in complex systems and modeling social behavior this seemed like a good choice. The other upside is that, I think, it will make for a much better 20 minute presentation. Explaining the "what" and the "why" of category theory in 20 minutes is hard.

I'm still in the process of learning this material. The meat of it is all in monoidal categories and these constructions on top of them called lenses and charts. Lenses compose system specifications—how they update and what variables they expose—while charts compose system behaviors like trajectories, steady states, and periodic orbits. Together they form the vertical and the horizontal compositions of the double category of arenas, a thing I don't actually understand. Because of this,

and because I'm running out of time for research, I'll stick to lenses for the project and rely on simulation to peer into the systems' behaviors.

As a sample of the work done for that, here's my definition for the category of arenas and lenses over a cartesian category $\mathcal{C}$.

Lenses, Categorically

Given a cartesian category, it's possible to construct a new category of systems whose states are drawn from the objects of your base category and whose rules for updating their state are drawn from the morphism of your base category. We call this category $\mathbf{Lens}_{\mathcal{C}}$, where $\mathcal{C}$ is the base category. The objects in this category are called arenas and the morphisms between them are called lenses.

Definition 3 (Lenses)**.** Given a cartesian category $\mathcal{C}$ and objects $A^-, A^+, B^-, B^+ \in ob(\mathcal{C})$, a lens consists of a passforward map $f : A^+ \to B^+$ and a passback map $f^\# : A^+ \times B^- \to A^-$ between two arenas as follows:

$$\begin{pmatrix} f^\# \\ f \end{pmatrix} : \begin{pmatrix} A^- \\ A^+ \end{pmatrix} \leftrightarrows \begin{pmatrix} B^- \\ B^+ \end{pmatrix}$$

For our purposes we'll be sticking to two such categories: the category of lenses over the category of sets and functions, $\mathbf{Lens}_{\mathbf{Set}}$, for discrete systems and the category of Euclidean spaces and smooth functions, $\mathbf{Lens}_{\mathbf{Euc}}$, for differential systems.

Definition 4 (The Category $\mathbf{Lens}_{\mathcal{C}}$)**.** Given the cartesian category $\mathcal{C}$, the category $\mathbf{Lens}_{\mathcal{C}}$ has the following properties.

1. A collection of objects called arenas. An arena $\begin{pmatrix} A^- \\ A^+ \end{pmatrix}$ is a pair of objects in $\mathcal{C}$.
2. For each pair of arenas a collection of morphisms $\begin{pmatrix} f^\# \\ f \end{pmatrix} : \begin{pmatrix} A^- \\ A^+ \end{pmatrix} \leftrightarrows \begin{pmatrix} B^- \\ B^+ \end{pmatrix}$ called lenses.
3. For each arena an identity lens $\begin{pmatrix} \pi_2 \\ id_{A^+} \end{pmatrix} : \begin{pmatrix} A^- \\ A^+ \end{pmatrix} \leftrightarrows \begin{pmatrix} A^- \\ A^+ \end{pmatrix}$ where the passback map π_2 is the projection $\pi_2 : A^+ \times A^- \to A^-$.
4. For any two compatible lenses a composite lens as follows:

$$\begin{pmatrix} f^\# \\ f \end{pmatrix} : \begin{pmatrix} A^- \\ A^+ \end{pmatrix} \leftrightarrows \begin{pmatrix} B^- \\ B^+ \end{pmatrix}$$

$$\begin{pmatrix} g^\# \\ g \end{pmatrix} : \begin{pmatrix} B^- \\ B^+ \end{pmatrix} \leftrightarrows \begin{pmatrix} C^- \\ C^+ \end{pmatrix}$$

$$\begin{pmatrix} g^\# \\ g \end{pmatrix} \circ \begin{pmatrix} f^\# \\ f \end{pmatrix} : \begin{pmatrix} A^- \\ A^+ \end{pmatrix} \leftrightarrows \begin{pmatrix} C^- \\ C^+ \end{pmatrix}$$

   such that the passforward map is defined as $a^+ \mapsto g(f(a^+))$, and the passback map is defined as $(a^+, c^-) \mapsto f^\#(a^+, g^\#(f(a^+), c^-))$.

$\mathbf{Lens}_{\mathcal{C}}$ is also a monoidal category with the monoidal product being defined as follows.

1. The monoidal unit is the arena $\begin{pmatrix} 1 \\ 1 \end{pmatrix}$.
2. Given lenses $\begin{pmatrix} f^\# \\ f \end{pmatrix} : \begin{pmatrix} A^- \\ A^+ \end{pmatrix} \leftrightarrows \begin{pmatrix} B^- \\ B^+ \end{pmatrix}$ and $\begin{pmatrix} g^\# \\ g \end{pmatrix} : \begin{pmatrix} C^- \\ C^+ \end{pmatrix} \leftrightarrows \begin{pmatrix} D^- \\ D^+ \end{pmatrix}$, the monoidal product is $\begin{pmatrix} f^\# \\ f \end{pmatrix} \otimes \begin{pmatrix} g^\# \\ g \end{pmatrix} : \begin{pmatrix} A^- \times C^- \\ A^+ \times C^+ \end{pmatrix} \leftrightarrows \begin{pmatrix} B^- \times D^- \\ B^+ \times D^+ \end{pmatrix}$ such that the passforward map is defined as $(a^+, c^+) \mapsto (f(a^+), g(c^+))$ and the passback map is defined as $((a^+, c^+), (b^-, d^-)) \mapsto ((f^\#(a^+, b^-)), g^\#(c^+, d^-))$.

One thing to note here is that I'm specifying these maps in terms of elements. This is okay since the objects I'll be using are all sets.[1]

Lenses, Computationally

Since I'll be relying on simulation I wrote some code specifying all the necessary bits to wire lenses together. It's not tested yet. I haven't written the code to actually wire up and simulate the systems yet. The code I do have is below.

[1] Apparently it's okay anyway but the reasoning gets very technical and I don't fully understand it yet.

```
'''
Code to support wiring lenses together.
- Lens dataclass
- Composition of lenses
- Monoidal product of lenses
'''

from dataclasses import dataclass


@dataclass
class Lens:
    update: callable
    expose: callable


#lens composition
def compose(A: Lens, B: Lens) -> Lens:
    up = lambda a: A.update(a[0], B.update(A.expose(a[0]), a[1]))
    out = lambda a: B.expose(A.expose(a))
    return Lens(up, out)


#lens monoidal product
def monoidal_product(A: Lens, B: Lens) -> Lens:
    up = lambda ac, bd: (A.update(ac[0], bd[0]), B.update(ac[1], bd[1]))
    out = lambda ac: (A.expose(ac[0]), B.expose(ac[1]))
    return Lens(up, out)



def main() -> None:
    return


if __name__ == "__main__":
    main()
```

How It's Going and What's Left

The project has gone well. I've read a lot. I've gone through a lot of scratch paper covered in diagrams. And I've laid face down on the floor with my eyes closed a lot trying to visualize representable functors and the Yoneda lemma. The process has been much different than a lecture class, but it's also been different than my summer research project where I struggled multiple times with overwhelm. It's honestly felt almost effortlessly effortful, like how I normally research and learn about topics that interest me only more and faster: read, take notes, think about it while I walk the kids to the park, and write out summaries of my understanding and check them against the source. It's been fun. It's been a lot of time and work, but it's been fun.

The most difficult thing so far has been grappling with just how much material there is and making those choices of what I do or do not cover in the paper. My paper is already 13 pages long and most of that is definitions.

I obviously still have work to do on the paper and I haven't started work on the presentation at all. For research this term, I still have to figure out what systems I want to build up and build them. I also need to choose a couple of simple ones to test my code against first. And that's it. I should be able to get all of it done before the term ends.