Communication Through Category Theory

Kenneth Watkins

June 17, 2022

Contents

1 Part 1 **2**
- 1.1 Defining a Category 2
- 1.2 Haskell Types and the Category of Set 2
- 1.3 Examples of Categories 3
- 1.4 Subcategories and Haskell Typeclasses 5
- 1.5 Monoids 5

1 Part 1

1.1 Defining a Category

A Category consist of two things ...

- Objects
- Arrows known as morphisms

Category Theory serves as an interpretation for the foundation of mathematics much like Set Theory and Type Theory. The key focus is on the the composition of mathematical structures. These structures are called objects. Again the focus is on the composition of the objects which is captured with the notion of an arrow (formally a morphism) that points to objects.

A Category C is made up of objects and arrows called morphism for which the follow properties hold true.

- **Identity** - Objects must have a an arrow originating from itself to itself as the Identity morphism. This arrow serves as a unit of composition such that when composed with any Arrow that either starts at A or ends at A it gives back the same arrow. So if f is an arrow then $f \circ id_A = f$
- **Composition** - Given the objects A, B, C and two arrows $f = A \to B$ and $g = B \to C$, there must exist a third arrow from A to C represented as $g \circ f = A \to C$.
- **Associative** - Given 3 arrows f, g, h, composition must be associative the they must be associative $h \circ (g \circ f) = (h \circ g) \circ f = h \circ g \circ f$

1.2 Haskell Types and the Category of Set

A type can be thought of as a set of values. As an example the Haskell type Bool is a two element set of value True and False. Sets can be finite or infinite. In Haskell x :: Integer is saying x is an element of Integer. The category of sets is called Set. Its special because we can peak in at its objects.

In the category of set **objects are sets** and **morphisms are functions between sets**

- empty set has no elements
- there are special one element sets
- functions map elements of one set to elements of another set
- functions can map two elements to one but not one element to two
- there exist an identity function that maps each element of a set to itself

In the Haskell implementation the category of Haskell types and functions is referred to as Hask. By forgetting the bottom in non terminating functions we can treat Hask is the Category Set

Functions with multiple type arguments have two interpretations as well. $a \to a \to a$ can be interpreted as a function that takes multiple arguments and the last value is the return type or it can be interpreted as $a \to (a \to a)$ function that takes an argument and returns a function requiring another argument.

Figure 1: Translation of Category of Set to Haskell ...

Set	Haskell Type	Description
Empty Set	Void	Type not inhibited by any values
Singleton Set	() "Unit"	Type has one value that always exist
Two Element Set	Bool	true or false, functions to this type are called predicates

1.3 Examples of Categories

Free Construction is the process of completing a given structure by extending it with the minimum number of items to satisfy its laws. Graphs are the free constructor for the Free category. chains of length zero serve as the identity morphism in the free category.

Thin categories are those where Hom-set is the set of morphisms from object a to object b in the category of C written as C(a,b) or HomC(a,b)

A hom-set is the set of morphisms from object a to object b in the category of C written as $C(a, b)$ or $Hom_C(a, b)$ Every hom-set is either empty, a singleton, or a preorder. Preorders can have cycles where as partial orders cannot

Sorting algorithms like quick-sort, bubble sort, merge sort, only work on total orders. Partial orders use topological sorts there is at most one morphism going from any object to any other object.

Figure 2: Examples of Categories ...

Category	Objects	Morphisms	Description
Empty	None	None	A category without objects or morphisms
Set	Sets	Functions	A category in which we can peek into the objects
Graph (Free Category)	Nodes that represent objects	all possible chains of composable edges	A category for Free construction
Preorder (Thin Category)	general non specific objects	relationship between objects where objects $\leq$	only one morphism between two objects
Partial Order	general non specific objects	relationship between objects where objects are $\leq$ and that if $a \leq b$ and $b \leq a$ then a must be the same as b	only one morphism between two objects
Linear/Total Order	general non specific objects	relationship between objects where objects are $\leq$ and that if $a \leq b$ and $b \leq a$ then a must be the same as b. Finally it states any two objects are in relation to one another	only one morphism between two objects

1.4 Subcategories and Haskell Typeclasses

In the category of C a subcategory of C is a sub-collection of the objects of C and a sub-collection of the morphism of C which map only between objects in the sub-collection of the subcategory

Subcategories can be represented as typeclasses in Haskell and have two interpretations depending on the amount of type parameters

- A single parameter typeclass can be interpreted as describing the members of a set of types
- Multi-parameter typeclasses is interpreted as a relation on types

Type class can be used to defined a collection of types that are members of a particular class. A type signature with a universally quantified type variable constrained by the typeclass

Example type signatures and their interpretation

- `Eq a => a`

  the collections of types that are members of the class Eq. Members of Eq are a sub-collection of objects in the Category of Hask

- `Eq a, Eq b => (a -> b)`

  is a sub-collection of the morphisms in Hask mapping between objects in the sub-collection of objects defined in Eq

1.5 Monoids

Monoids are a mathematical concept found across different branches of mathematics.

In set theory a monoid is a set equipped with a binary function that is associative and a unit element. An example is addition on the set of integers is equipped with the pseudo function $(+) :: a \to a \to a$.

Monoid M is a set with a unit element e and binary operation. Such that if $a, b, c \in M$ then $\ldots$

$$a \circ b \in M$$

$$(a \circ b) \circ c = a \circ (b \circ c)$$

$$e \circ a = a \circ e = a$$

In category theory a monoid is a one object category with a set of morphisms that follow the rules of composition.

Given a single object category C. C's object is c, C's morphism $c \to c$. C's unit is $1_c : c- > c$

Operations on M are exactly the same as operations on $Morph(C)$ allowing us to regard the category C as the monoid M. The correspondence is reversible

$$f \circ g \in Morph(C)$$

$$(f \circ g) \circ h = f \circ (g \circ h)$$

$$1_c \circ f = f \circ 1_c = f$$

given the category of $C = c$ we obtain a monoid M whose elements are arrows of C This allows us to extract a set from a single object category.

because $c \in C$ and the hom-set of c is $Hom_C(c, c)$ resulting in a Set monoid M whose elements are morphisms of C

In the Hask we can define a subcategory definition for the Monoid using a typeclass.

```
class Monoid m where
mempty  :: m
mappend :: m -> m -> m
```