Reference Sheet for Elementary Category Theory

Categories

A **category** $\mathcal{C}$ consists of a collection of “objects” `Obj` $\mathcal{C}$, a collection of “morphisms” `Mor` $\mathcal{C}$, an operation `Id` associating a morphism $\mathtt{Id}_a$ `: a → a` to each object `a`, a parallel pair of functions `src, tgt : Mor` $\mathcal{C}$ `→ Obj` $\mathcal{C}$, and a “composition” operation `_⨾_ : ∀{A B C : Obj} → (A → B) → (B → C) → (A → C)` where for objects `X` and `Y` we define the *type* `X → Y` as follows

$$f : X \to Y \quad \equiv \quad \mathsf{src}\ f = X \ \wedge\ \mathsf{tgt}\ f = Y \qquad \text{defn-Type}$$

Moreover composition is required to be associative with `Id` as identity.
Instead of `src` and `tgt` we can instead assume primitive a ternary relation `_:_→_` and regain the operations precisely when the relation is functional in its last two arguments:

$$f : A \to B \ \wedge\ f : A' \to B' \implies A = A' \ \wedge\ B = B' \qquad \text{unique-Type}$$

When this condition is dropped, we obtain a *pre-category*; e.g., the familiar *Sets* is a pre-category that is usually treated as a category by making morphisms contain the information about their source and target: `(A, f, B) : A → B` rather than just `f`. *This is sometimes easier to give, then src and tgt! C.f. Alg(F).*

A categorical statement is an expression built from notations for objects, typing, morphisms, composition, and identities by means of the usual logical connectives and quantifications and equality.

Even when morphisms are functions, the objects need not be sets: Sometimes the objects are *operations* –with an appropriate definition of typing for the functions. The categories of F-algebras are an example of this.

Example Categories.

- Each digraph determines a category: The objects are the nodes and the paths are the morphisms typed with their starting and ending node. Composition is catenation of paths and identity is the empty path.
- Each preorder determines a category: The objects are the elements and there is a morphism `a → b` named, say, `(a, b)`, precisely when *ab*.

Functors

A **functor** $F : \mathcal{A} \to \mathcal{B}$ is a pair of mappings, denoted by one name, from the objects, and morphisms, of $\mathcal{A}$ to those of $\mathcal{B}$ such that it respects the categorical structure:

$$F\,f : F\,A \to_{\mathcal{B}} F\,B \quad \Leftarrow \quad f : A \to_{\mathcal{A}} B \qquad \text{functor-Type}$$

$$F\,\mathsf{Id}_A \ = \ \mathsf{Id}_{F\,A} \qquad \text{Functor}$$

$$F\,(f ⨾ g) \ = \ F\,f ⨾ F\,g \qquad \text{Functor}$$

The two axioms are equivalent to the single statement that *functors distribute over finite compositions, with* `Id` *being the empty composition*

$$F(f ⨾ \cdots ⨾ g) \ = \ F\,f ⨾ \cdots ⨾ F\,g$$

Use of Functors.

- In the definition of a category, “objects” are “just things” for which no internal structure is observable by categorical means –composition, identities, morphisms, typing.
  Functors form the tool to deal with “structured” objects
  Indeed in $\mathcal{S}et$ the aspect of a structure is that it has “constituents”, and that it is possible to apply a function to all the individual constituents; this is done by $F\,f : F\,A \to F\,B$.
- For example, let $A = A \times A$ and $f = (x, y)\ \ (f\,x, f\,y)$. So is or represents the structure of pairs; A is the set of pairs of A, and f is the function that applies f to each constituent of a pair.
  - A *binary operation on* A is then just a function $A \to A$; in the same sense we obtain “F-ary operations”.
- Also, *Seq* is or represents the structure of sequences; *Seq* A is the structure of sequences over A, and *Seq* f is the function that applies f to each constituent of a sequence.
- Even though $F\,A$ is still just an object, a thing with no observable internal structure, the functor properties enable to exploit the “structure” of $F\,A$ by allowing us to “apply” a f to each “constituent” by using $F\,f$.

Category $\mathcal{A}lg(F)$

- For a functor $F : \mathcal{A} \to \mathcal{D}$, this category has “F-algebras”, F-ary operations in $\mathcal{D}$ as, objects – i.e., objects are $\mathcal{D}$-arrows $F\,A \to A$ – and F-homomorphisms as morphisms, and it inherits composition and identities from $\mathcal{D}$.

  $$f : \oplus \to_F \otimes \quad \equiv \quad \oplus ⨾ f = F\,f ⨾ \otimes \qquad \text{defn-Homomorphism}$$

  $$\mathsf{Id} : \oplus \to_F \oplus \qquad \text{id-Homomorphism}$$

  $$f ⨾ g : \oplus \to_F \odot \quad \Leftarrow \quad f : \oplus \to_F \otimes \ \wedge\ g : \otimes \to_F \odot \qquad \text{comp-Homomorhism}$$

  Note that category axiom (unique-Type) is not fulfilled since a function can be a homomorphism between several distinct operations. However, we pretend it is a category in the way discussed earlier, and so the carrier of an algebra is fully determined by the operation itself, so that the operation itself can be considered the algebra.

  Theorem (comp-Homomorhism) *renders a semantic property as a syntactic condition!*

- A **contravariant functor** $\mathcal{C} \to \mathcal{D}$ is just a functor $\mathcal{C}^{op} \to \mathcal{D}^{op}$.
- A **bifunctor** from $\mathcal{C}$ to $\mathcal{D}$ is just a functor $\mathcal{C}^{\,2} \to \mathcal{D}$.

Naturality

A natural transformation is nothing but a structure preserving map between functors. “Structure preservation” makes sense, here, since we’ve seen already that a functor is, or represents, a structure that objects might have.

Further Reads

- Roland Backhouse
- Grant Malcolm

- ◇ Lambert Meertens
- ◇ Jaap van der Woude