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` $\to$ `a` to each object `a`, a parallel pair of functions `src, tgt :  Mor` $\mathcal{C} \to$ `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:

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

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

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

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

$$F(f \mathbin{⨾} \cdots \mathbin{⨾} g) \ = \ F\, f \mathbin{⨾} \cdots \mathbin{⨾} F\, g$$

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 → F B.*
- For example, let *A* = *A* × *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 → 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 $\mathcal{D}$.

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

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

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

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.

As discussed before for the case $F : \mathcal{C} \to \mathcal{S}et$, each $F\ A$ denotes a structured set and F denotes the structure itself.

For example, is the structure of pairs, *Seq* is the structure of sequences, *Seq* the structure of pairs of sequences, *Seq Seq* the structure of sequences of sequences, and so on.

A “transformation” from structure F to structure G is a family of functions $\eta : \forall\{A\} \to F\ A \to G\ A$; and it is “natural” if each η_a doesn’t affect the *constituents* of the structured elements in $F\ A$ but only reshapes the structure of the elements, from an F-structure to a G-structure.

Reshaping the structure by η commutes with subjecting the constituents to an arbitrary morphism.

$$\eta : F \dot{\to} G \quad \equiv \quad \forall f \bullet F\ f \,;\, \eta_{\mathsf{tgt}\ f} \ = \ \eta_{\mathsf{src}\ f} \,;\, G\ f \qquad \text{ntrf-Def}$$

This is ‘naturally’ remembered: Morphism $\eta_{\mathsf{tgt}\ f}$ has type $F(\mathsf{tgt}\ f) \to G(\mathsf{tgt}\ f)$ and therefore appears at the target side of an occurrence of f; similarly $\eta_{\mathsf{src}\ f}$ occurs at the source side of an f. *Moreover* since η is a transformation *from* F to G, functor F occurs at the source side of an η and functor G at the target side.

- One also says η_a *is natural in* its parameter a.
- If we take $G = \mathsf{Id}$, then natural transformations $F \dot{\to} \mathsf{Id}$ are precisely F-homomorphisms.
- Indeed, a natural transformation is a sort-of homomorphism in that the image of a morphism after reshaping is the same as the reshaping of the image.

Example natrual transformations
- *rev : Seq $\dot{\to}$ Seq : [a_1, ..., a_n] [a_n, ..., a_1]* reverses its argument thereby reshaping a sequence structure into a sequence structure without affecting the constituents.
- *inits : Seq $\dot{\to}$ Seq Seq : [a_1, ..., a_n] [[], [a_1], $\cdots$, [a_1, ..., a_n]]* yields all initial parts of its argument thereby reshaping a sequence structure into a sequence of sequences structure, not affecting the constituents of its argument.

$$J\eta : JF \dot{\to} JG \quad \Leftarrow \quad \eta : F \dot{\to} G \quad \text{where } (J\eta)_A \coloneqq J(\eta_A) \qquad \text{ntr-Ftr}$$

$$\eta K : FK \dot{\to} GK \quad \Leftarrow \quad \eta : F \dot{\to} G \quad \text{where } (\eta K)_A \coloneqq \eta_{(K\ A)} \qquad \text{ntr-Poly}$$

$$\mathsf{Id}_F : F \dot{\to} F \quad \text{where } (\mathsf{Id}_F)_A \coloneqq \mathsf{Id}_{(F\ A)} \qquad \text{ntrf-Id}$$

$$\epsilon \,;\, \eta : F \dot{\to} H \quad \Leftarrow \quad \epsilon : F \dot{\to} G \ \wedge \ \eta : G \dot{\to} H$$
$$\text{where } (\epsilon \,;\, \eta)_A = \epsilon_A \,;\, \eta_A \qquad \text{ntrf-Compose}$$

Category $\mathcal{F}unc(\mathcal{C},\ \mathcal{D})$ consists of functors $\mathcal{C} \to \mathcal{D}$ as objects and natrual transformations between them as objects. The identity transformation is indeed an identity for transformation composition, which is associative.

Heuristic To prove $\phi = \phi_1 \,;\, \cdots \,;\, \phi_n : F \dot{\to} G$ is a natural transformation, it suffices to show that each ϕ is a natural transformation.
- Theorem (ntrf-Compose) renders proofs of semantic properties to be trivial type checking!
- E.g., It’s trivial to prove *tails = rev ; inits ; Seq rev* is a natural transformation by type checking, but to prove the naturality equation by using the naturality equations of *rev* and *inits* –no definitions required– necessitates more writing, and worse: Actual thought!

Adjunctions

An adjunction is a particular one-one correspondence between different kinds of morphisms in different categories.

An **adjunction** consists of two functors $L : \mathcal{A} \to \mathcal{B}$ and $R : \mathcal{B} \to \mathcal{A}$, as well as two (not necessarily natural!) transformations $\eta : \mathsf{Id} \to RL$ and $\epsilon : LR \to \mathsf{Id}$ such that

$$f = \eta_A \,;\, Rg \qquad \equiv \qquad Lf \,;\, \epsilon_B = g$$
$$\text{where } f : A \to_{\mathcal{A}} RB \text{ and } g : LA \to_{\mathcal{B}} B \qquad \text{Adjunction}$$

Reading right-to-left: In the equation $Lf \,;\, \epsilon_B = g$ there is a unique solution to the unknown f. Dually for the other direction.

That is, *each L-algebra g is uniquely determined –as an L-map followed by an ϵ-reduce– by its restriction to the adjunction’s unit η.*

A famous example is “Free ⊣ Forgetful”, e.g. to *define* lists for which the above becomes: Homomorphisms on lists are uniquely determined, as a map followed by a reduce, by its restriction to the singleton sequences.

We may call f the restriction, or lowering, of g to the “unital case” and write $f = \lfloor g \rfloor = \eta_A \,;\, Rg$. Also, we may call g the extension, or raising, of f to an L-homomorphism and write $g = \lceil f \rceil = Lf \,;\, \epsilon_B$. The above equivalence now reads:

$$f = \lfloor g \rfloor \qquad \equiv \qquad \lceil f \rceil = g \qquad \text{Adjunction-Inverse}$$

$$\lfloor g \rfloor_{A,B} = \eta_A \,;\, Rg \ : \ A \to_{\mathcal{A}} RB \text{ where } g : LA \to_{\mathcal{B}} B \qquad \text{lad-Type}$$

$$\lceil f \rceil_{A,B} = Lf \,;\, \epsilon_B \ : \ LA \to_{\mathcal{B}} B \text{ where } f : A \to_{\mathcal{A}} RB \qquad \text{rad-Type}$$

Note that $\lceil$ is like ‘r’ and the argument to $\lceil\rceil$ must involve the R-ight adjoint in its type; {**L**}ad takes morphisms involving the {**L**}eft adjoint ;)

This equivalence expresses that ‘lad’ $\lfloor\rfloor$, from *l*eft *ad*jungate, and ‘rad’ $\lceil\rceil$, from *r*ight *ad*jungate, are each other’s inverses and constitute a correspondence between certain morphisms. *Being a bijective pair, lad and rad are injective, surjective, and undo one another.*

We may think of $\mathcal{B}$ as having all complicated problems so we abstract away some difficulties by *r*aising up to a cleaner, simpler, domain via rad $\lceil\rceil$; we then solve our problem there, then go back *down* to the more complicated concrete issue via $\lfloor\rfloor$, lad. (E.g., $\mathcal{B}$ is the category of monoids, and $\mathcal{A}$ is the category of sets; L is list functor.)

The η and ϵ determine each other and they are *natural* transformations. ntrf-Adj

(“zig-zag laws”) The unit has a post-inverse while the counit has a pre-inverse:

$$\mathsf{Id} = \eta \,;\, R\epsilon \qquad \text{unit-Inverse}$$

$$\mathsf{Id} = L\eta \,;\, \epsilon \qquad \text{Inverse-counit}$$

The unit and counit can be regained from the adjunction inverses,

$$\eta = \lfloor \mathsf{Id} \rfloor \qquad \text{unit-Def}$$

$\epsilon = \lceil \mathsf{Id} \rceil$ counit-Def

Lad and rad themselves are solutions to the problems of interest, (Adjunction).

$L\lfloor g \rfloor \,;\, \epsilon = g$ lad-Self

$\eta \,;\, R\lceil f \rceil = f$ rad-Self

The following laws assert a kind of monoic-ness for ϵ and a kind of epic-ness for η. Pragmatically they allow us to prove an equality by shifting to a possibly easier equality obligation.

$\eta \,;\, Rg = \eta \,;\, Rg' \quad \equiv \quad g = g'$ lad-Unique

$Lf \,;\, \epsilon = Lf' \,;\, \epsilon \quad \equiv \quad f = f'$ rad-Unique

Lad and rad are natural transformations in the category $\mathcal{F}unc(\mathcal{A}^{op} \times \mathcal{B}, \mathcal{S}et)$ realising $(LX \to Y) \cong (X \to GY)$ where X, Y are the first and second projection functors and $(- \to -) : \mathcal{C}^{op} \times \mathcal{C} \to \mathcal{S}et$ is the hom-functor such that $(f \to g)h = f \,;\, h \,;\, g$.
By extensionality in $\mathcal{S}et$, their naturality amounts to the laws:

$\lfloor Lx \,;\, g \,;\, y \rfloor \;=\; x \,;\, \lfloor g \rfloor \,;\, Ry$ lad-Fusion

$\lceil x \,;\, f \,;\, Ry \rceil \;=\; Lx \,;\, \lceil f \rceil \,;\, y$ rad-Fusion

Also,
- ◇ Left adjoints preserve colimits such as initial objects and sums.
- ◇ Right adjoints preserve limits such as terminal objects and products.

More

Nice Stuff

Further Reads

- ◇ Roland Backhouse
- ◇ Grant Malcolm
- ◇ Lambert Meertens
- ◇ Jaap van der Woude
- ◇ *Adjunctions* by Fokkinga and Meertens