Introduction to Category Theory*
OPLSS 2023

Daniela Petrisan

June. 28th to July 1st, 2023

Contents

1 Categories and functors **2**
1.1 Categories . . . 2
1.1.1 Examples . . . 3
1.1.2 Free category on a graph . . . 4
1.2 Isomorphisms . . . 4
1.3 Functors . . . 5
1.3.1 Examples . . . 5
1.3.2 Covariance and contravariance . . . 6
1.3.3 Subcategories . . . 7

2 Automata **8**

3 Natural transformations **12**

4 Limits and colimits **14**
4.1 Instances of limits and colimits . . . 14
4.2 Limits and colimits . . . 17
4.2.1 Alternative definitions . . . 20

5 Adjunctions **21**

6 Monads and Algebras **25**

*Transcribed by Noah Bertram, Zak Sines, Mateo Torres-Ruiz, Andrey Yao

1 Categories and functors

A category consists of objects, arrows between them, and a way to compose arrows together. As it turns out, this kind of structure is pervasive throughout mathematics, providing the ability to view seemingly completely separate mathematical structures through the same lens. We now make this statement precise by defining a category and providing a plethora of examples.

1.1 Categories

Definition 1 (Category). A *category* $\mathcal{C}$ consists of the following data:

- a class of *objects* $A, B, C, \ldots$,
- a class of *arrows* (also called *morphisms*) $f, g, h, \ldots$, where for each arrow f there are objects $dom(f)$ and $cod(f)$, respectively called the *domain* and *codomain* of f. We write $f : A \to B$ whenever $A = dom(f)$ and $B = cod(f)$,
- a partial *composition* map, defined as $g \circ f : A \to C$ for any two arrows $f : A \to B$ and $g : B \to C$ where $dom(g) = cod(f)$,
- for each object A in $\mathcal{C}$, an *identity arrow* $1_A : A \to A$,

subject to the following axioms:

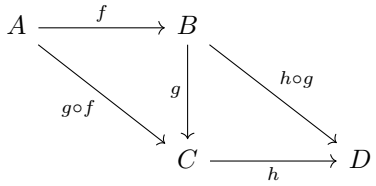
- *associativity*: given arrows $f : A \to B, g : B \to C, h : C \to D$, we have

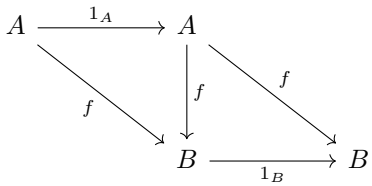
$$(h \circ g) \circ f = h \circ (g \circ f)$$

- *identity laws*: for each $f : A \to B$,

$$f \circ 1_A = f = 1_B \circ f$$

We will often make use of *diagrammatic reasoning* to talk about commutativity of arrows in our categories. The edges of these diagrams will denote objects of our category, while the arrows will be morphisms. We can diagrammatically express the two category axioms as





Note 1. Note that our definition of a category refers to objects and morphisms each creating a class and not a set. This can be explained by recalling Russell's paradox. If objects formed a set, it would be impossible to talk about a category of sets Set, given that no set can contain all sets.

Notation 1. There are multiple ways of refering to the collection of morphisms between two objects within a category $\mathcal{C}$: $\mathcal{C}(A, B)$, $Hom_C(A, B)$ or simply $Hom(A, B)$ all refer to the collection of arrows from A to B.

Category	Objects	Arrows
Set	Sets	Functions
Rel	Sets	Relations
$(X, \leq)$	$x \in X$	$x \to y \iff x \leq y$
$(M, \cdot, 1_M)$	$*$	$m \in M$
Mon	Monoids	Monoid homomorphisms
Top	Topological spaces	Continuous maps
$\mathsf{Vec}_{\mathbb{K}}$	Vector spaces	Linear maps
Pre	Pre-orders	Monotone maps

1.1.1 Examples

The following are examples of categories.

Definition 2 (Preorder). A set X together with a relation $\leq \subseteq X \times X$ is a *preorder* if $x \leq x$ for all $x \in X$ ($\langle X, \leq \rangle$ is reflexive) and $x \leq y$ and $y \leq z$ implies $x \leq z$ for all $x, y, z \in X$ (this is, $\langle X, \leq \rangle$ is transitive).

Example 1 (Preorders). It is not hard to see that any category in which for every pair of objects there exists a unique morphism between them can be seen as a preorder and, similarly, any preorder $\langle X, \leq \rangle$ forms a category in which the objects are the elements of X and there is a morphism between $x, y \in X$ whenever $x \leq y$.

Definition 3 (Monoid). A *monoid*, $(M, \cdot_M, 1_M)$, is a set M, equipped with a binary operation $\cdot_M : M \times M \to M$ and an identity element $1_M \in M$ satisfying

- $(m_1 \cdot_M m_2) \cdot_M m_3 = m_1 \cdot_M (m_2 \cdot_M m_3)$ for all $m_1, m_2, m_3 \in M$
- $m \cdot_M 1_M = 1_M \cdot_M m = m$ for all $m \in M$.

Example 2 (Matrices as a monoid). For each natural number n, the set $\mathrm{Mat}_{n,n}(\mathbb{Z})$ of $n \times n$ matrices with integer-valued entries forms a monoid under matrix multiplication. Note that matrix multiplication is associative, which satisfies the corresponding monoid law. The monoid identity in this case is the identity matrix I, with 1 on the diagonal and 0 on all other entries. Formally, the monoid is $(\mathrm{Mat}_{n,n}(\mathbb{Z}), \cdot, I)$.

Example 3 (Integers as a monoid). The set of integers $\mathbb{Z}$ also forms a monoid under integer multiplication. The monoid identity is the integer 1. In this case, since integer multiplication is commutative, we say $(\mathbb{Z}, \times, 1)$ is a commutative monoid.

Definition 4 (Monoid homomorphism). Recall that a *monoid homomorphism* is a function $f : M \to N$ between $(M, \cdot_M, 1_M), (N, \cdot_N, 1_N)$ monoids, such that $f(m \cdot_M n) = f(m) \cdot_N f(n)$ for each $m, n \in M$ and $h(1_M) = 1_N$. Informally, a monoid homomorphism "preserves" the monoid structure by mapping identities to identities, and by preserving the monoid products.

Example 4 (Determinant as monoid homomorphism). Using the two example monoids above, recall that the determinant is a map $\det : \mathrm{Mat}_{n,n}(\mathbb{Z}) \to \mathbb{Z}$ satisfying the properties that for two matrices A, B, $\det(A \cdot B) = \det(A) \det(B)$. Since the determinant of the identity matrix I is 1, we can conclude that det is actually a monoid homomorphism.

Example 5 (Monoid as a one object category). We can equivalently think of a monoid as a category with just one object, $*$, where the arrows are the elements of the monoid, the identity the unit element 1_M of M, our monoid, and the composition the binary operation $\cdot_M$.

Example 6 (Category of monoids). We can take a further step and consider Mon, the category that has monoids as its objects and monoid homomorphisms as its arrows. It is easy to check that composition of monoid homomorphisms is again a monoid homomorphism, and that such composition is associative.

1.1.2 Free category on a graph

Let $G = (V, E)$ be a directed graph. The free category on G, denoted $\mathcal{C}(G)$ has as objects the elements of V, with morphisms being the set of paths on the vertices of G. That is, given any path,

$$v_0 \xrightarrow{e_1} v_1 \xrightarrow{e_2} \cdots \xrightarrow{e_n} v_n$$

we consider

$$e_1 e_2 \cdots e_n : v_0 \to v_n$$

a morphism in $\mathcal{C}(G)$, where the empty path on $v \in V$ provides the identity $1_v : v \to v$, and where composition is given by path concatenation.

Exercise Verify that path concatenation in a graph is associative, completing the construction of $\mathcal{C}(G)$.

1.2 Isomorphisms

We briefly construct a notion of sameness amongst objects in a category.

Definition 5 (Isomorphism). In a category $\mathscr{C}$, a morphism $f : X \to Y$ is an *isomorphism* if there exists a morphism $g : Y \to X$ such that $g \circ f = 1_X$ and $f \circ g = 1_Y$.

The morphism g in the definition above, may referred to as the inverse of f. Let us see some examples.

Example 7 (Isomorphisms). The following are examples of isomorphisms in various categories:

- In Set, the isomorphisms are the bijective functions.
- In Grp the category of groups, the isomorphisms are the bijective group homomorphisms.
- In Pre the category of preorders and monotone maps, the isomorphisms are bijective monotone maps whose inverses are also monotone.
- In Cat the category of small categories with functors as morphisms, the isomorphisms are functors $F : \mathscr{C} \to \mathscr{D}$ s.t. there exist $F : \mathscr{D} \to \mathscr{C}$ where the composition both ways yields the identity functors on $F : \mathscr{C}$ and $\mathscr{D}$.
- In $[\mathscr{C}, \mathscr{D}]$, the functor category, the isomorphisms are the natural isomorphisms. A natural transformation $\alpha : F \Rightarrow G$ is a natural isomorphism if each component α_X for object X in $\mathscr{C}$ is an isomorphism.

Definition 6 (Isomorphic). We say two objects X, Y are isomorphic, denoted $X \cong Y$, if there is an isomorphism from X to Y.

1.3 Functors

Following the intuition from previous examples, the most important thing about our categories is not the objects but the morphisms. It is then natural to ask, what then is a good notion for morphisms between categories themselves. The answer to this will be functors, which capture the notion of arrows between categories.

Definition 7 (Functor). Let $\mathcal{C}, \mathcal{D}$ be categories. A functor $F : \mathcal{C} \to \mathcal{D}$

- associates to each object A in $\mathcal{C}$ an object FA in $\mathcal{D}$,
- to each morphism $f : A \to B$ in $\mathcal{C}$ a morphism $Ff : FA \to FB$,

subject to the following axioms:

- $F(f \circ g) = F(f) \circ F(g)$ whenever f and g compose in $\mathcal{C}$,
- $F(1_A) = 1_{FA}$ whenever A is in $\mathcal{C}$.

1.3.1 Examples

Example 8 (Forgetful functor). Consider the category of monoids Mon. Each of the objects in this category has an underlying set giving the elements of a particular monoid. We can "forget" the monoid structure on this category and recover the underlying sets through a *forgetful functor* that acts on objects of Mon by only taking its underlying set (and *forgetting* the monoid's composition operation and unit), and on arrows $f : (M, \cdot_M, 1_M) \to (N, \cdot_N, 1_N)$ by recovering a mapping between the two underlying sets,

$$\begin{aligned} U : \mathsf{Mon} &\to \mathsf{Set} \\ (M, \cdot_M, 1_M) &\mapsto M \\ f : (M, \cdot_M, 1_M) \to (N, \cdot_N, 1_N) &\mapsto f : M \to N \end{aligned}$$

Example 9 (Free monoid functor). This time, we will define a functor F that goes the opposite direction of the forgetful functor above. For any set $X \in \mathsf{Set}$, we can define A^* the *free monoid over* A as the set of all formal finite length (possibly empty) strings, with characters taken from the elements of A. Formally, $A^* = \bigcup_{i=0}^{\infty} \{x_1 x_2 \dots x_i | x_1, x_2, \dots, x_i \in A\}$. The monoid composition operation $\cdot_{A^*}$ is string concatenation, which is easily checked as associative. The monoid unit 1_{A^*} is the empty string, which is a two-sided unit with regards to string concatenation. For example, if $A = \{a, b, c, d, e\}$, then $addb, cac \in A^*$, and $addb \cdot_{A^*} cac = addbcac$.

The functor F acts on morphisms in Set, which are just functions between sets, in the following way. For each function $f : A \to B$, $F(f) : A^* \to B^*$ is a function that takes any string $x_1 \dots x_n \in A^*$ and sends it to the string of the same length $f(x_1) \dots f(x_n) \in B^*$. Obviously, this sends the empty string to the empty string. It can also be shown easily that it commutes with string concatenation $\cdot_{A^*}$. Thus, $F(f)$ is a monoid homomorphism.

It remains to check that F follows the functor laws. For each set A and the identity function id_A, $F(id_A)$ obviously sends each string in A^* to itself, so $F(f) = id_{A^*}$, the identity morphism on the monoid A^*. It is also straightforward to show that F preserves function compositions.

To summarize, we have the free functor:

$$\begin{aligned} F : \mathsf{Set} &\to \mathsf{Mon} \\ A &\mapsto (A^*, \cdot_{A^*}, 1_{A^*}) \\ f : A \to B &\mapsto F(f) : (A^*, \cdot_{A^*}, 1_{A^*}) \to (B^*, \cdot_{B^*}, 1_{B^*}) \end{aligned}$$

Note 2. We will later see that the free and forgetful functors between Set and Mon above together form an instance of a concept called "adjunction", and those two functors are called adjoint functors. There are many other free-forgetful pairs between other categories that also form adjunctions.

Example 10 (Preorder categories and monotone maps). Let $(X, \leq)$ and $(Y, \leq)$ be two preorders. We can view $(X, \leq)$ as a category $\mathscr{X}$ where the objects are the elements of X, and for any $x_1, x_2 \in X$, there is a morphism $x_1 \to x_2$ in the category iff $x_1 \leq x_2$. Since preorders are reflexive, all elements x has $x \leq x$, so every object has an identity morphism. The transitivity of preorders means that composition of morphism is well-defined, as if $f : x_1 \to x_2$ and $g : x_2 \to x_3$ are morphisms, then $g \circ f$ corresponds to that $x_1 \leq x_3$. Similarly, we can view $\mathscr{Y}$ as a category. Note that in preorder categories, for each pair of objects there is at most one morphism between them.

We show that functors from $\mathscr{X}$ to $\mathscr{Y}$ are *just* monotone maps. Take any functor $F : \mathscr{X} \to \mathscr{Y}$. The action of F on the objects of the category makes it exactly a function: $X \to Y$. Whenever $x_1 \leq x_2 \in X$, there exists morphism $f : x_1 \to x_2$ in $\mathscr{X}$ by definition. Since functors preserve domains and codomains of morphisms, $F(f)$ is a morphism $F(x_1) \to F(x_2)$ in $\mathscr{Y}$, but the existence of such morphism means that $F(x_1) \leq F(x_2)$ in Y. We have thus shown that F is monotonic as a function $X \to Y$. Conversely, for any monotone map $F : X \to Y$, we can view it as a functor $\mathscr{X} \to \mathscr{Y}$. Define F's actions on morphisms by sending each $f : x_1 \to x_2$, which means $x_1 \leq x_2$, to the morphism $F : F(x_1) \to F(x_2)$, which exists because of monotonicity, and is unique by definition of preorder categories. This uniqueness then ensures that F commutes with composition of morphisms, and that it preserves identity morphisms.

Example 11 (Monoids as one-object categories and monoid homomorphisms as functors). Recall from Example 1 that monoids can be seen as one-object categories. Let $(M, \cdot_M, 1_M)$ and $(N, \cdot_N, 1_N)$ be two monoids, and let the one-object categories $\mathscr{M}$, $\mathscr{N}$ be their categorifications. We claim that functors from $\mathscr{M}$ to $\mathscr{N}$ are *just* monoid homomorphisms from M to N. Note that in this setting, since both categories have a single object each, all morphisms in each category can compose with each other, and we can disregard the action of functors on objects.

Given a monoid homomorphism $f : M \to N$, we can use the same function as the functor's action on the morphisms. Conversely, given each functor $F : \mathscr{M} \to \mathscr{N}$ we can view its action on morphisms as a function from $\mathscr{M} \to \mathscr{N}$. It's quite easy to see that the functor laws match the definition of monoid homomorphism quite directly.

1.3.2 Covariance and contravariance

Sometimes you may run into something that morally should be a functor but isn't due to composition not behaving quite right:

$$F(f \circ g) = Fg \circ Ff. \tag{1}$$

This can be corrected by considering the *opposite category*. Given a category $\mathscr{C}$, the *opposite category* $\mathscr{C}^{\mathrm{op}}$ has objects the objects of $\mathscr{C}$ and morphisms $\mathscr{C}^{\mathrm{op}}(X, Y) = \mathscr{C}(Y, X)$, that is, $X \xrightarrow{f} Y$

in $\mathscr{C}$ is $Y \xrightarrow{f} X$ in $\mathscr{C}^{\mathrm{op}}$, with composition $g \circ^{\mathrm{op}} f = f \circ g$. Visually, $\mathscr{C}^{\mathrm{op}}$ just flips all the arrows of $\mathscr{C}$. Then, a functor-like map from $\mathscr{C}$ to $\mathscr{D}$ that satisfies (1) is actually a functor $\mathscr{C}^{\mathrm{op}} \to D$: given $g \in \mathscr{C}(X, Y)$ and $f \in \mathscr{C}(Y, Z)$, then $f \in \mathscr{C}^{\mathrm{op}}(Z, Y)$ and $g \in \mathscr{C}^{\mathrm{op}}(Y, X)$. Thus

$$F(g \circ^{\mathrm{op}} f) = F(f \circ g) = Fg \circ Ff.$$

We would refer to F as a *contravariant functor*. One can think of contravariance as just flipping the direction of the arrows. We also say a functor is *covariant* if it is a normal functor, i.e. the direction of the arrows stays the same.

1.3.3 Subcategories

Before continuing, we provide the following definitions.

Definition 8. A *subcategory* A of C is a category whose objects are all objects of C and whose morphisms are all morphisms of C.

Definition 9. A subcategory A of C is *full* if every morphism in C between objects that are also in A is also a morphism in A.

Given a subcategory A of C, we always have the *inclusion functor* $i : A \to C$, which sends $X \overset{i}{\mapsto} X$ and $(X \xrightarrow{f} Y) \overset{i}{\mapsto} (X \xrightarrow{f} Y)$.

Example 12. The category FinSet has finite sets as objects and functions between them as morphisms. This clearly is a full subcategory of Set.

2 Automata

Throughout, fix a finite set Σ. First we recall the following forms of automata:

Definition 10 (Deterministic automata). A *deterministic automaton* is a tuple $(Q, \delta, i \in Q, f \subseteq Q)$, where Q is a set and $\delta : \Sigma \times Q \to Q$. Here, Q is the set of states, i is the initial state, f is the set of accepting states, and δ is the transition function, where each state has exactly one transition for each letter in Σ. The *path* of word $w \in \Sigma^*$ in the automaton is the sequence

$$i \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$$

where $w = a_1 a_2 \cdots a_n$, and $\delta_{a_1}(i) = q_1$, $\delta_{a_2}(q_1) = q_2$, ..., $\delta_{a_n}(q_{n-1}) = q_n$, for $q_1, q_2, \ldots, q_n \in Q$. We say the automaton accepts w if in the path of w, $q_n \in f$. Then the *language* accepted by the automaton, $\mathcal{L} \subseteq \Sigma^*$, is the set of words accepted by the automaton.

Definition 11 (Non-deterministic automata). A *non-deterministic automaton* is a tuple $(Q, \delta, i \subseteq Q, f \subseteq Q)$, where Q is a set and $\delta : \Sigma \times Q \to 2^Q$. Again, Q is the set of states, i are the initial states, f is the set of accepting states, and δ is the transition function, which now has possibly many transitions for a given element. A *path* of word $w \in \Sigma^*$ in the automaton is a sequence

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$$

where $w = a_1 a_2 \cdots a_n$, and $q_1 \in \delta_{a_1}(q_0)$, $q_2 \in \delta_{a_2}(q_1)$, ..., $q_n \in \delta_{a_n}(q_{n-1})$, for $q_1, q_2, \ldots, q_n \in Q$. We say the automaton accepts w if for some path of w, $q_n \in f$ where q_n is the final state in the path. Then the *language* accepted by the automaton, $\mathcal{L} \subseteq \Sigma^*$, is the set of words accepted by the automaton.

For both deterministic and non-deterministic automata, the language accepted by the automaton is the subset of words that are accepted by the automaton. The definition of a language is relaxed in the case of *weighted* automata.

Definition 12. A *weighted* automaton over a field $\mathbb{K}$, is a tuple (Q, δ, I, F) , where Q is a set, $\delta : \Sigma \times Q \to (Q \to \mathbb{K})$ is the transition function, assigning for each element of Σ and Q, weights in $\mathbb{K}$ for each element of Q, $I : Q \to \mathbb{K}$, and $F : Q \to \mathbb{K}$, assigning initial and final weights to the states Q respectively. The language of a weighted automaton, A is a function $\mathcal{L} : \Sigma^* \to \mathbb{K}$ given as follows: Given a word $w \in \Sigma^*$, a path for w in A is a sequence of states

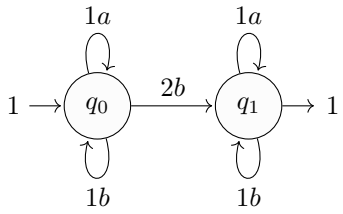
$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} q_n$$

where $w = a_1 a_2 \cdots a_n$. Call this path p. The weight of this path, denoted $\mathbf{w}(p)$, is

$$I(q_0) \cdot \delta_{a_1}(q_0)(q_1) \cdot \delta_{a_2}(q_1)(q_2) \cdots \delta_{a_n}(q_{n-1})(q_n) \cdot F(q_n).$$

Then $\mathcal{L}(w) = \sum_{p \in P_w} \mathbf{w}(p)$. Observe that when either $\{q \in Q | I(q) \neq 0\}$ or $\{q \in Q | F(q) \neq 0\}$ is finite (which occurs in particular when Q is finite), then the sum is finite.

Example 13. As an example, suppose that $\Sigma = \{a, b\}$ and consider the following weighted automaton over $\mathbb{Q}$:

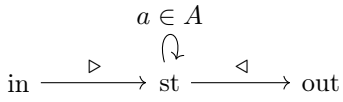


This automaton exhibits the language $\mathcal{L}(w) = 2|w|_b$, where $|w|_b$ is the number of occurrences of b in w. All paths with non-zero value start at q_0 and end at q_1.

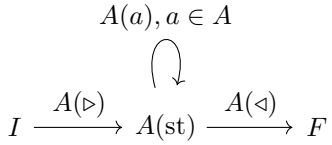
Now suppose that Q is finite, and that $|Q| = n$. Then we can equivalently define a weighted automaton as (I, μ, T) where $I \in \mathbb{K}^{1\times n}$, $T \in \mathbb{K}^{n\times 1}$, and $\mu : \Sigma \to \mathbb{K}^{n\times n}$, where the language of A for $w \in \Sigma^*$ is given by $I\mu(a_1) \cdot \mu(a_2) \cdots \mu(a_n)T$, where $w = a_1 a_2 \cdots a_n$.

Exercise Verify this equivalent definition.

We would like to arrange the data for the automata above in a categorical way. Consider the following graph:



Denote by $\mathscr{I}$ the free category on this graph. Observe that $\mathscr{I}(\text{in}, \text{st}) = \{\triangleright w | w \in \Sigma^*\}$ and $\mathscr{I}(\text{st}, \text{st}) \cong \Sigma^*$, that is, we can identify every morphism from st to st with a word from Σ. Now let $\mathscr{C}$ be a category and I, F two objects of $\mathscr{C}$. Then a $(\mathscr{C}, I, F)$-*automaton* is a functor $A : \mathscr{I} \to \mathscr{C}$ such that $A(\text{in}) = I$ and $A(\text{out}) = F$. The graph above becomes



Now let $\mathcal{O}$ be the full subcategory of $\mathscr{I}$ consisting of the objects in and out. Then the morphisms of this category are those of the form $\triangleright w \triangleleft : \text{in} \to \text{out}$ for $w \in \Sigma^*$. We refer to $\mathcal{O}$ as the *observational category*. Then a $(\mathscr{C}, I, F)$-*language* is a functor $\mathcal{L} : \mathcal{O} \to \mathscr{C}$ such that $\mathcal{L}(\text{in}) = I$ and $\mathcal{L}(\text{out}) = F$. Given an automaton $A : \mathscr{I} \to \mathscr{C}$, the language *accepted* by A is the composition $A \circ i : \mathcal{O} \to \mathscr{C}$ where $i : \mathcal{O} \to \mathscr{I}$ is the inclusion functor.

Let's consider the following examples:

Example 14. If we set $\mathscr{C} = \mathsf{Set}$, $I = \{*\}$, and $F = \{\top, \bot\}$, we obtain a *deterministic* automaton over A. In this case,

- $A(\text{in}) = \{*\}$

- $A(\text{out}) = \{\top, \bot\}$
- $Q := A(\text{st})$ becomes the set of states
- $i := A(\triangleright) : \{*\} \to Q$ becomes the choice of initial state
- $f := A(\triangleleft) : Q \to \{\top, \bot\}$ becomes the choice of final states
- $\delta_a := A(a) : Q \to Q$ becomes the transitions for the element $a \in \Sigma$

Then the graph above becomes

$$\begin{array}{ccccc} & & \delta_a, a \in A & & \\ & & \circlearrowright & & \\ \{*\} & \xrightarrow{\quad i \quad} & Q & \xrightarrow{\quad f \quad} & \{\top, \bot\} \end{array}$$

Then the language accepted by this automaton takes the morphism $\triangleright w \triangleleft : \text{in} \to \text{out}$ to a function $A(\triangleright w \triangleleft) : \{*\} \to \{\top, \bot\}$. In particular, there are exactly two functions $\{*\} \to \{\top, \bot\}$, namely, $* \mapsto \top$ and $* \mapsto \bot$, the former corresponding to accepting and the latter to rejecting. Thus A *accepts* w if $A(\triangleright w \triangleleft)(*) = \top$ and *rejects* w if $A(\triangleright w \triangleleft)(*) = \bot$.

Example 15. If we set $\mathscr{C} = \mathsf{Rel}$, and $I = F = \{*\}$ we obtain a *non-deterministic* automaton over A. This is similar to the case for the deterministic automaton, but we relax functions to be relations, allowing for multiple transitions. In this case,

- $A(\text{in}) = \{*\}$
- $A(\text{out}) = \{*\}$
- $Q := A(\text{st})$ becomes the set of states
- $i := A(\triangleright) : \{*\} \nrightarrow Q$ becomes the choice of initial states
- $f := A(\triangleleft) : Q \nrightarrow \{*\}$ becomes the choice of final states
- $\delta_a := A(a) : Q \nrightarrow Q$ becomes the (possibly many) transitions for the element $a \in \Sigma$

Then the graph above becomes

$$\begin{array}{ccccc} & & \delta_a, a \in A & & \\ & & \circlearrowright & & \\ \{*\} & \xrightarrow{\quad i \quad} & Q & \xrightarrow{\quad f \quad} & \{*\} \end{array}$$

Then the language accepted by this automaton takes the morphism $\triangleright w \triangleleft : \text{in} \to \text{out}$ to a relation $A(\triangleright w \triangleleft) : \{*\} \nrightarrow \{*\}$. In particular, there are exactly two relations $\{*\} \nrightarrow \{*\}$, namely, $\{(*, *)\}$ and $\emptyset$, the former corresponding to accepting and the latter to rejecting. Thus A *accepts* w if $A(\triangleright w \triangleleft) = \{(*, *)\}$ and *rejects* w if $A(\triangleright w \triangleleft) = \emptyset$.

Example 16. If we set $\mathscr{C} = \mathsf{Vec}_{\mathbb{K}}$, and $I = F = \mathbb{K}$, we obtain a *weighted* automaton over A.
In this case,

- $A(\mathrm{in}) = \mathbb{K}$
- $A(\mathrm{out}) = \mathbb{K}$
- $Q := A(\mathrm{st})$ becomes a vector space spanned by the states
- $i := A(\triangleright) : \mathbb{K} \to Q$ becomes the linear map providing the choice of initial weights when evaluated at 1
- $f := A(\triangleleft) : Q \to \mathbb{K}$ becomes the linear map of final weights when evaluated at 1
- $\delta_a := A(a) : Q \to Q$ becomes the linear map providing transition weights for the element $a \in \Sigma$

Then the graph above becomes

$$\begin{array}{ccccc} & & \delta_a, a \in A & & \\ & & \circlearrowright & & \\ \mathbb{K} & \xrightarrow{\quad i \quad} & Q & \xrightarrow{\quad f \quad} & \mathbb{K} \end{array}$$

Then the language accepted by this automaton takes the morphism $\triangleright w \triangleleft : \mathrm{in} \to \mathrm{out}$ to a linear map $A(\triangleright w \triangleleft) : \mathbb{K} \to \mathbb{K}$. Recall that classically, the language of a weighted automaton is a map $\mathcal{L} : \Sigma^* \to \mathbb{K}$. This can be recovered by evaluating $A(\triangleright w \triangleleft)$ at 1, that is, $\mathcal{L}(w) = A(\triangleright w \triangleleft)(1)$.

3 Natural transformations

Just like objects have arrows between them, we have seen that categories also have arrows in the form of functors. It is no surprise then that we can put arrows between our functors, resulting in what we call *natural transformations*.

Definition 13 (Natural transformation). Let $\mathcal{C}, \mathcal{D}$ be two categories and $F, G : \mathcal{C} \to \mathcal{D}$ two functors between them. A natural transformation from F to G, denoted $\alpha : F \Rightarrow G$, consists of a family of morphisms,

$$(\alpha_C : FC \to GC)_{C \in Obj(\mathcal{C})}$$

such that for every morphism $f : C \to C'$ in $\mathcal{C}$, the following diagram commutes

$$\begin{array}{ccc} FC & \xrightarrow{\alpha_C} & GC \\ {\scriptstyle Ff}\downarrow & & \downarrow{\scriptstyle Gf} \\ FC' & \xrightarrow[\alpha_{C'}]{} & GC' \end{array}$$

Algebraically,

$$(Gf \circ \alpha_C = \alpha_{C'} \circ Ff)$$

Notation 2. Diagrammatically, we write natural transformations as vertical arrows between functors. For a natural transformation $\alpha : F \Rightarrow G$ we have

$$\begin{array}{ccc} & F & \\ \mathcal{C} & \Downarrow \alpha & \mathcal{D} \\ & G & \end{array}$$

Given $\beta : G \Rightarrow H$, their composition $\beta \circ \alpha : F \Rightarrow H$ is

$$\begin{array}{ccc} & F & \\ \mathcal{C} & \Downarrow \alpha & \mathcal{D} \\ & \Downarrow \beta \; G & \\ & H & \end{array} \quad \equiv \quad \begin{array}{ccc} & F & \\ \mathcal{C} & \Downarrow \beta \circ \alpha & \mathcal{D} \\ & H & \end{array}$$

This is

$$\begin{array}{ccccc} FC & & \xrightarrow{(\beta \circ \alpha)_C} & & HC \\ & {\scriptstyle \alpha_C} \searrow & & \nearrow {\scriptstyle \beta_C} & \\ & & GC & & \end{array}$$

Definition 14 (Functor category). Let $\mathscr{C}, \mathscr{D}$ be categories. Then $[\mathscr{C}, \mathscr{D}]$ is a category where the objects are the functors from $\mathscr{C}$ to $\mathscr{D}$ and the arrows are natural transformations between the functors.

Exercise 1. Verify that this is a category by checking that composition, often referred to as vertical composition, is associative.

Example 17 (Functor category $[\mathscr{M}, \mathsf{Set}]$). Let $(M, \cdot_M, 1_M)$ be a monoid. We can view M as a one-object category $\mathscr{M}$ with objects $\{*\}$. It follows that the functor category $[\mathscr{M}, \mathsf{Set}]$ has the following properties. Each object of the $[\mathscr{M}, \mathsf{Set}]$, which will just be a functor $F : \mathscr{M} \to \mathsf{Set}$, has to send $*$ to some set X, and for all morphism m in $\mathscr{M}$, which is just an element of M, $F(m)$ is a set function from $F(*) = X$ to itself, with $F(1_m) = id_X$. Thus, F is a left monoid action on the set X.

Exercise 2. Let F, G be functors from $\mathscr{M}$ to Set, where $\mathscr{M}$ is a monoid M viewed as a category. Show that morphisms $\alpha : F \Rightarrow G$ in $[\mathscr{M}, \mathsf{Set}]$, which are just natural transformations from F to G, corresponds to M-action homomorphisms.

Exercise 3 (Graph homomorphisms as natural transformations). A directed graph $G = (E, V)$, where $E \subseteq V \times V$, can defined as a functor $F : \mathscr{C} \to \mathsf{Set}$ in the following way. Define $\mathscr{C}$ to be this category below (identity morphisms are omitted):

$$0 \underset{t}{\overset{s}{\rightrightarrows}} 1$$

Where we define F to have:

$F(0) \triangleq E$	The edge set
$F(1) \triangleq V$	The vertex set
$F(s) : E \to V$	The "source" function
$F(t) : E \to V$	The "target" function

A *graph homomorphism* between two direct graphs $G = (V, E)$ and $G' = (V', E')$ is a function $f : V \to V$ such that for all $(v1, v_2) \in E$, $(f(v_1), f(v_2)) \in E'$ as well. Show that graph homomorphisms correspond to natural transformations between these "graphs as functors" as defined above.

4 Limits and colimits

4.1 Instances of limits and colimits

Before formally defining limits and colimits, which are constructs defined by some "universal property", we introduce common instances of these limits and colimits and illustrate them using concrete examples.

Definition 15 (Initial object). An object X in a category $\mathscr{C}$ is *initial* if for every object Y in $\mathscr{C}$, there exists a unique morphism $! : X \to Y$.

Example 18 (Initial object). In Set, the empty set $\varnothing$ is initial because for any set X, there is exactly one function from $\varnothing$ to X, namely the empty function.

In Mon, the monoid with one element $(\{\text{“OPLSS”}\}, \cdot, \text{“OPLSS”})$, or if you like, the isomorphic copy $(\{\text{“SAM”}\}, \cdot, \text{“SAM”})$, is the initial object.

Example 19 (Category without initial object).

$$A \underset{g}{\overset{f}{\rightrightarrows}} B$$

The category generated by this diagram does not have an initial object. A cannot be initial because there are two morphisms from A to B, violating uniqueness. B cannot be initial because there is no morphism from B to A, violating existence.

Theorem 1. If X and Y are two initial objects in a category $\mathscr{C}$, then $X \cong Y$ via a unique isomorphism.

Proof. By definition there exists unique morphisms $X \xrightarrow{!_X} Y$ and $Y \xrightarrow{!_Y} X$, then composition provides arrows

$$X \xrightarrow{!_Y \circ !_X} X \quad \text{and} \quad Y \xrightarrow{!_X \circ !_Y} Y.$$

Because both X and Y are initial, both exhibit unique arrows to themselves, and moreover must be 1_X and 1_Y respectively. This says that

$$!_Y \circ !_X = 1_X \quad \text{and} \quad !_X \circ !_Y = 1_Y,$$

giving us the desired isomorphism. Lastly, because $!_X$ and $!_Y$ are unique, this isomorphism is unique. □

Exercise 4. In the category $\mathsf{Auto}(L)$ for some language $\mathcal{L}$, show that the following object is initial:

$$1 \xrightarrow{\epsilon} A^* \xrightarrow{\mathcal{L}?} 2 \qquad \{w \mapsto aw\}_a \text{ (loop on } A^*\text{)}$$

Here, ϵ is the constant function that sends the unique element of 1 to the empty string in A^*. The function $\mathcal{L}?$ takes any word in A^* and returns true if and only if the word is in $\mathcal{L}$.

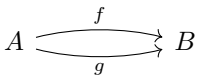
Definition 16 (Terminal object). An object X in a category $\mathscr{C}$ is *terminal* if for every object Y in $\mathscr{C}$, there exists a unique morphism $! : Y \to X$.

Exercise 5. Show that terminal objects are isomorphic via a unique isomorphism.

Example 20 (Terminal object). The following are examples of terminal objects:

- In Set, any singleton set, i.e. $\{\text{“Sam”}\}$ is a terminal object.
- in Mon, the trivial monoid of one element is both initial and terminal. When this occurs, the object is commonly referred to by either the *zero object* or the *null object*.

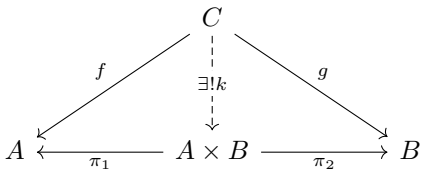
Example 21 (No terminal objects).



This category from before doesn't have terminal objects, for similar reasons to why it doesn't have an initial object.

Definition 17 (Product). Let $\mathscr{C}$ be a category, and let A and B be two objects in $\mathscr{C}$. A product of A and B is a tuple $(A \times B, \pi_1, \pi_2)$ where $A \times B$ is an object of $\mathscr{C}$, $\pi_1 : A \times B \to A$, $\pi_2 : A \times B \to B$, such that for any object C and morphisms $f : C \to A$ and $g : C \to B$, there exists a unique morphism $k : C \to A \times B$ such that $\pi_1 \circ k = f$ and $\pi_2 \circ k = g$.

We often refer to the maps π_1 and π_2 as projections of $A \times B$, and typically we omit mention of them, just referring to $A \times B$ as the product. We can represent the product of A and B in the following commutative diagram:

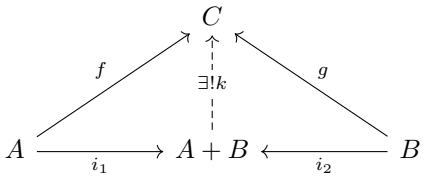


Example 22 (Product). Examples of products:

- In Set, products are cartesian products of sets. The unique morphism $k : C \to A \times B$, given functions f, g from set C, sends each c to $(f(c), g(c))$.
- In Mon, a product of two monoids $(M, \cdot_M, 1_M)$ and $(N, \cdot_N, 1_N)$ has object the monoid $(M \times N, \cdot, (1_M, 1_N))$, where $\cdot$ is component-wise multiplication. The projections are obvious.
- More generally, in categories that are “algebraic structures” over Set defined by generators and equations, such as Grp, $\mathsf{BoolAlg}$, Ab, Ring, etc., the product of two objects is defined using the product of the underlying sets, and the algebraic operations are defined component-wise.
- Viewing a preorder $(X, \leq)$ as a category $\mathscr{X}$, the product of two elements $x, y \in X$ viewed as objects of $\mathscr{X}$ is just the meet $x \wedge y$, if it exists.

The coproduct is dual to the product. Like the product, we often refer to it with just the object, having i_1 and i_2 implicit. We can represent the coproduct of A and B in the following commutative diagram:

Definition 18 (Coproduct). Let $\mathscr{C}$ be a category, and let A and B be two objects in $\mathscr{C}$. A product of A and B is a tuple $(A \times B, i_1, i_2)$ where $A + B$ is an object of $\mathscr{C}$, $i_1 : A \to A + B$, $i_2 : B \to B + A$, such that for any object C and morphisms $f : A \to C$ and $g : B \to C$, there exists a unique morphism $k : A + B \to C$ such that $k \circ i_1 = f$ and $k \circ i_2 = g$.



Example 23 (Coproducts). Examples of coproducts:

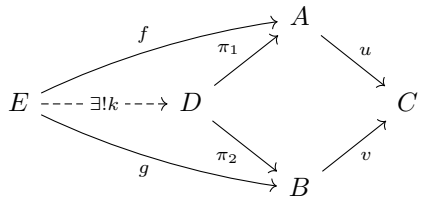
- In Set, coproducts are disjoint unions of sets.
- In Mon, coproducts are the *free products* of monoids.

Exercise 6. Show that products are coproducts, if they exist, are unique up to a unique isomorphism.

Definition 19 (Pullback). Let $\mathscr{C}$ be a category and let A, B, and C be objects in C, along with morphisms $u : A \to C$ and $v : B \to C$. A *pullback* of u and v is the tuple $(D, \pi_1 : D \to A, \pi_2 : D \to B)$ satisfying $u \circ \pi_1 = v \circ \pi_2$, and for which the following condition holds: for all objects E of $\mathscr{C}$ and morphisms $f : E \to A$, $g : E \to B$ such that $u \circ f = v \circ g$, there exists unique morphism $k : E \to D$ such that

$$u \circ \pi_1 \circ k = u \circ f = v \circ g = v \circ \pi_2 \circ k.$$

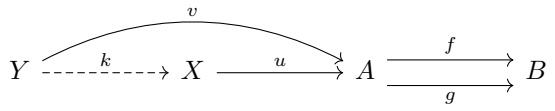
The pullback can be expressed by the following commutative diagram:



As we had with the initial object and product, there is a dual notion for pullback.

Exercise 7. Define the dual notion of the pullback, known as a *pushout* (pushforward).

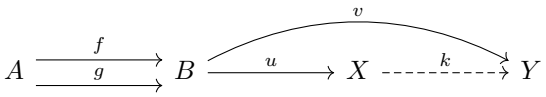
Definition 20 (Equalizer). Consider two morphisms $f, g : A \to B$ in a category $\mathcal{C}$. An *equalizer* of f and g is an object X together with an arrow $u : X \to A$ such that $f \circ u = g \circ u$ and for all $v : Y \to A$ there exists a unique $k : Y \to X$,



Example 24 (Equalizer in Set). Let us consider what an equalizer is in the category Set. For two morphisms $f, g : A \to B$, an equalizer X is a set $\{a \in A : f(a) = g(a)\}$ with $u : X \to A$ the inclusion function. This is, an equalizer in Set gives us the set of all solutions to the equation $f(a) = g(a)$.

Similar to previous instances of limits, we can get the colimit verison of what an equalizer is by considering an equalizer in the opposite category.

Definition 21 (Coequalizer). Given two morphisms $f, g : A \to B$ in a category $\mathcal{C}$, a *coequalizer* is an object X together with an arrow $u : B \to X$ such that $u \circ f = u \circ g$ and for all $v : B \to Y$ there exists a unique $k : X \to Y$,



Before introducing an example of a concrete coequalizer, let us recall the concept of an equivalence class and how these quotient a set.

Definition 22 (Equivalence class). Given an equivalence relation $\sim$, we define the *equivalence class* $[x]$ of $x \in X$ as

$$[x] = \{y \in X : x \sim y\}$$

and we say that the equivalence classes of X form a *partition* of X.

Definition 23 (Quotient). Given a set X together with an equivalence relation $\sim$ on it, we define the *quotient* of X by $\sim$ as the set of all equivalence classes on X, denoted by $X/\sim$.

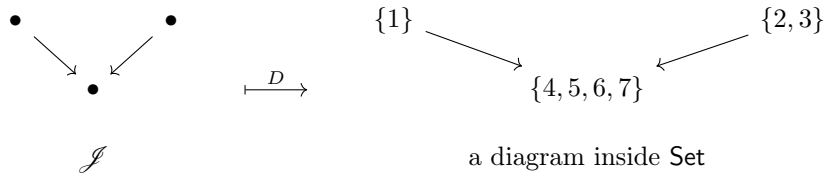
Example 25 (Coequalizer in Set). Given two morphisms $f, g : A \to B$ in Set, a coequalizer X is the set $\{f(a) \sim g(a) : a \in A\}$ with $u : A \to X$ the quotient map.

4.2 Limits and colimits

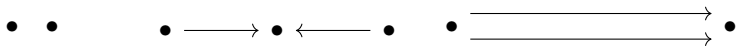
One may be able to notice that all these instances of limits have an associated diagram that seeming provides all the data. We can indeed take advantage of our categorical language to formalize these diagrams as well. Before this, we say that a category is *small* if the class of objects and morphisms forms a set.

Definition 24 (Diagram). A diagram on a category $\mathscr{C}$ is a functor $D : \mathscr{J} \to \mathscr{C}$, where $\mathscr{J}$ is a small category. In this case, we say D has *shape* $\mathscr{J}$.

Example 26 (Diagram). Here is an example of a diagram from $\mathscr{J}$ to Set, where $\mathscr{J}$ can be represented pictorially below on the left.



Here is the index category $\mathscr{J}$ for each of the (co)limit examples we saw before:

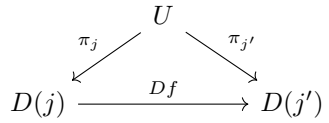


Initial & Terminal (Co)product Pullback (Co)equalizer

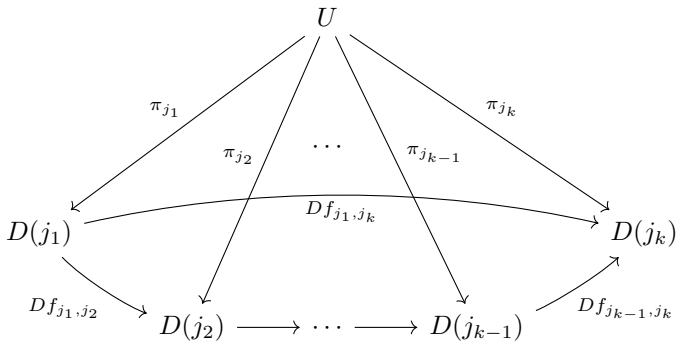
Note that the index category corresponding to the initial and terminal objects is the *empty category*, the category with no objects. As can be seen by the examples, the picture is not quite complete. Before we can complete it, we need to define a new functor. Given an object U of $\mathscr{C}$, let $\Delta_U : \mathscr{J} \to \mathscr{C}$ be the functor that sends all objects of $\mathscr{J}$ to U and all morphisms to the identity. This is known as the *constant functor*. Now we can complete the pictures.

Definition 25 (Cone). A cone over a diagram $D : \mathscr{J} \to \mathscr{C}$ is an object U of $\mathscr{C}$ together with a natural transformation $\pi : \Delta_U \Rightarrow D$.

Unpacking the definition, we obtain that for each object j of $\mathscr{J}$, there is a morphism $\pi_j : U \to D(j)$ that is natural in j. This exactly gives us the following commutative diagram for each morphism $j \xrightarrow{f} j'$:



To better see why they are called cones, consider the following not completely accurate diagram representing an instance of a cone:



Exercise 8. Draw the cones corresponding to the product, pullback, and equalizer.

We have almost arrived at the complete picture of a limit. If you recall the definitions for initial object, product, etc. above, you will notice that their definitions rely on having a unique morphism to the object whenever another object also has a similar commuting diagram. We can state this differently, saying that whenever we have another cone, the morphisms of that cone factor through the limiting cone. Note that since $\mathscr{J}$ is small, and, assuming that $\mathscr{C}$ is *locally small*, we have that $[\mathscr{J}, \mathscr{C}]$ is locally small. Then $[\mathscr{J}, \mathscr{C}](\Delta_U, D)$ can be considered a set which consists of the cones with object U.

Definition 26 (Limit). An object $\lim D$ in a locally small category C is a *limit of a diagram* $D : \mathscr{J} \to \mathscr{C}$ if there is a natural (contravariant) isomorphism

$$\mathscr{C}(V, \lim D) \cong [\mathscr{J}, \mathscr{C}](\Delta_V, D),$$

for every object V in $\mathscr{C}$.

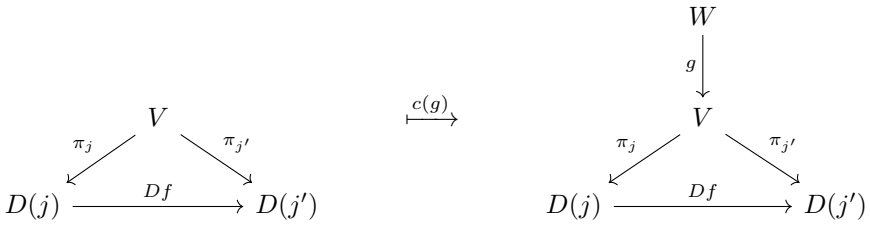
To elaborate, for $\lim D$ to be a limit means that for every cone $\pi : \Delta_V \to D$, there exists a unique morphism $f : V \to \lim D$ in $\mathcal{C}$ such that for every morphism $g : W \to V$ the following diagram commutes (in Set):

$$\begin{array}{ccc} \mathscr{C}(V, \lim D) & \xrightarrow{\sim} & [\mathscr{J}, \mathscr{C}](\Delta_V, D) \\ {\scriptstyle g^\#}\downarrow & & \downarrow{\scriptstyle c(g)} \\ \mathscr{C}(W, \lim D) & \xrightarrow{\sim} & [\mathscr{J}, \mathscr{C}](\Delta_W, D) \end{array}$$

Here, $\mathscr{C}(V, \lim D) \xrightarrow{g^\#} \mathscr{C}(W, \lim D)$ is the morphism given by *precomposition of* g, that is

$$(V \xrightarrow{f} \lim D) \xmapsto{g^\#} (W \xrightarrow{g} V \xrightarrow{f} \lim D),$$

and $c(g)$ is the function



The isomorphism is really a set bijection, meaning that there is a one-to-one correspondence between diagrams with V as an object and morphisms from V to $\lim D$. But how do we recover the complete diagrams, seen in the definition of product, equalizer, and pullback? We can recover the correct cone for $\lim D$ by using the isomorphism with $1_{\lim D} \in \mathscr{C}(\lim D, \lim D)$ to obtain a corresponding cone in $[\mathscr{J}, \mathscr{C}](\Delta_V, D)$; call this cone $\pi_{\lim D}$. Now to see that every cone factors through it, consider any cone $\pi_W : \Delta_W \Rightarrow D$. Then by the set isomorphism, there is a corresponding morphism $k : W \to \lim D$. Applying this morphism to the commuting square above, we obtain

$$\pi_W = c(\pi_{\lim D}),$$

which says that π_W factors through $\pi_{\lim D}$.

Where does this naturality notion come from? It is exactly the same condition that arises in that of a natural transformation. In fact, that is exactly what it lurking in the definition of a limit. The isomorphism of sets above comes from an isomorphism of functors. On the left, we have the functor $\mathscr{C}(-, \lim D) : \mathscr{C}^{\mathrm{op}} \to \mathsf{Set}$ which sends V to $\mathscr{C}(V, \lim D)$ and morphisms $f : X \to Y$ to $f^\# : \mathscr{C}(Y, \lim D) \to \mathscr{C}(X, \lim D)$ as defined above. The other functor sends an object V to the set $[\mathscr{J}, \mathscr{C}](\Delta_V, D)$, but the remaining part of the definition is left to the following exercise:

Exercise 9. Show that the naturality condition given in the definition of a limit, and elaborated on below it, arises from a natural isomorphism between two functors.

4.2.1 Alternative definitions

We now give alternative (but equivalent) definitions of limits and colimits[1]. The following definitions are commonly found in algebra texts and can be useful for grasping how limits and colimits of algebraic categories look like.

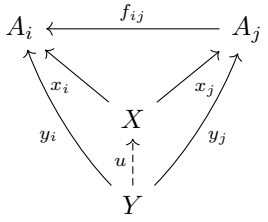
Definition 27 (Directed set). We say that a preordered set $\langle D, \leq \rangle$ is a directed set if every pair of elements has an upper bound.

Definition 28 (Inverse system). Given a directed set $\mathcal{I} = \langle D, \leq \rangle$, $(A_i)_{i \in \mathcal{I}}$ a family of objects in $\mathcal{I}$ and $f_{ij} : A_j \to A_i$ a family of morphisms (sometimes called the *transition morphisms*) for all $i \leq j$, we say that $\langle A_i, f_{ij} \rangle_{\mathcal{I}}$ forms an *inverse system* over $\mathcal{I}$.

Definition 29 (Inverse limit). We define the inverse limit of an inverse system $\langle A_i, f_{ij} \rangle_{\mathcal{I}}$ as

$$A = \varprojlim A_i = \{\bar{a} \in \prod_{i \in \mathcal{I}} A_i \ : \ a_i = f_{ij}(a_j) \text{ for all } i \leq j \text{ in } \mathcal{I}\}$$

Categorically, the (inverse) limit of a system $\langle A_i, f_{ij} \rangle_{\mathcal{I}}$ is an object X together with a family of morphisms $x_i : X \to A_i$ satisfying $x_i = f_{ij} \circ x_j$ for all $i \leq j$ (we call all such triangles *cones*) with the universal property that for every other pair (Y, y_i) there is a unique morphism $u : X \to Y$ such that the following diagram commutes:



It should be noted that this is precisely a *limit*. We can similarly define algebraic *colimits*.

Definition 30 (Direct system). Given a directed set $\mathcal{I} = \langle D, \leq \rangle$, $(A_i)_{i \in \mathcal{I}}$ a family of objects indexed by D and $f_{ij} : A_i \to A_j$ a morphism for every $i \leq j$ such that f_{ii} is the identity of A_i and $f_{ik} = f_{jk} \circ f_{ij}$ for all $i \leq j \leq k$, we say that $\langle A_i, f_{ij} \rangle$ forms a *direct system* over $\mathcal{I}$.

Definition 31 (Direct limit). We define the *direct limit* of a direct system $\langle A_i, f_{ij} \rangle_{\mathcal{I}}$ as

$$A = \varinjlim A_i = \bigsqcup_i A_i \Big/ \sim .$$

Where $\sim$ is an equivalence relation and $x_i \sim x_j$ for $x_i \in A_i, x_j \in A_j$ iff there is some $k \in I$ with $i \leq k$ and $j \leq k$ such that $f_{ik}(x_i) = f_{jk}(x_j)$.

Exercise 10. Similar to what we did with limits, relate the above with a diagram of a colimit.

[1] Note that these were not shown during lectures.

5 Adjunctions

Adjunctions are another way of relating two categories, $\mathscr{C}$ and $\mathscr{D}$ by means of a pair of related functors, $F : \mathscr{C} \to \mathscr{D}$ and $G : \mathscr{D} \to \mathscr{C}$. Stated rather loosely, an adjunction means that morphisms in $\mathscr{C}$ when sent to $\mathscr{D}$ by F as morphisms in $\mathscr{D}$ when sent to $\mathscr{C}$ by G. Throughout, we assume we are working with *locally small* categories, that is, categories where the morphisms form a set.

Definition 32 (Adjunction). Let $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{D} \to \mathcal{C}$ be two functors. We say that F is *left adjoint* to G and write $F \dashv G$ if there is an isomorphism $\mathcal{D}(FX, Y) \cong \mathcal{C}(X, GY)$ *natural* in X and Y,

$$\mathcal{C} \underset{G}{\overset{F}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathcal{D}$$

In this case, we also say that G is *right adjoint* to F.

Notation 3. We can also write an adjunction between the functors F and G as

$$\frac{FX \to Y \quad (\mathcal{D})}{X \to FY \quad (\mathcal{C})}$$

Since we specified that the categories are locally small, the isomorphism here lies in Set, meaning that there is a one-to-one correspondence between morphisms in $\mathscr{D}(FX, Y)$ and morphisms in $\mathscr{C}(X, GY)$. Drawn out,

$$\mathscr{D}(FX, Y) \underset{(-)_\sharp}{\overset{(-)^\sharp}{\rightleftarrows}} \mathscr{C}(X, GY)$$

More explicitly, given $f : FX \to Y$, we have $f^\sharp : X \to GY$, and given $g : X \to GY$, we get $g_\sharp : FX \to Y$:

$$f : FX \to X \rightsquigarrow f^\sharp : X \to GY$$
$$g : X \to GY \rightsquigarrow g_\sharp : FX \to Y$$

But what does it mean for the isomorphism to be natural in X and Y here? We will see that, loosely speaking, this naturality condition says that for two adjoint funtors $F : \mathcal{C} \to \mathcal{D}$, $G : \mathcal{D} \to \mathcal{C}$, $F \dashv G$, the morphisms $FX \to Y$ are essentially the same as the morphisms in $X \to GY$, with X in $\mathcal{C}$ and Y in $\mathcal{D}$.

We now elaborate on what it means to be natural in X and Y. For the isomorphism to be natural in X would mean that for any morphism $f : X \to X'$, the following diagram commutes

$$\begin{array}{ccc} \mathcal{D}(FX, Y) & \xrightarrow{(-)^\sharp} & \mathcal{C}(X, GY) \\ {\scriptstyle \mathcal{D}(Ff, Y)} \uparrow & & \uparrow {\scriptstyle \mathcal{C}(f, GY)} \\ \mathcal{D}(FX', Y) & \xrightarrow{(-)^\sharp} & \mathcal{C}(X', GY) \end{array}$$

where $\mathscr{D}(Ff, Y) : \mathscr{D}(FX', Y) \to \mathscr{D}(FX, Y)$ is the function given by *preimaging* morphisms from $\mathscr{D}(FX', Y)$ with $Ff : FX \to FX'$. Spelled out,

$$(FX' \xrightarrow{u} Y) \xrightarrow{\mathscr{D}(Ff,Y)} (FX \xrightarrow{Ff} FX' \xrightarrow{u} Y).$$

Similarly, $\mathscr{C}(f, GY)$ is a set function that preimages morphisms from $\mathscr{C}(X', GY)$ to obtain morphisms in $\mathscr{C}(X, GY)$. The diagram gives us the identity

$$u^\sharp \circ f = (u \circ Ff)^\sharp \tag{2}$$

for any $u : FX' \to Y$.

Being natural in Y is quite similar: for any morphism $g : Y \to Y'$, we have that

$$(g \circ v)_\sharp = Gg \circ v_\sharp \tag{3}$$

for any other morphism $v : FX \to Y$.

Exercise 11. Let $\mathbb{1}$ be the category with exactly one object $\{*\}$ and one morphism, 1_*.

1. Show that $\mathbb{1}$ is the terminal object in Cat.
2. If exists, what is a left adjoint (resp. right adjoint) to the unique functor $\mathscr{A} \xrightarrow{!} \mathbb{1}$ for each category $\mathscr{A}$?

Example 27. We have an adjunction between monoids and sets,

$$\mathsf{Set} \underset{U}{\overset{F}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathsf{Mon}$$

where U is the forgetful functor and F the free monoid functor. Indeed, we have that for a monoid $\mathbb{M} = (M, \circ, 1)$, it is the same to give a morphism in $\mathsf{Mon}(FX, \mathbb{M})$ than to give a function living in $\mathsf{Set}(X, U\mathbb{M})$.

Example 28.

$$\mathsf{Set} \underset{U}{\overset{F}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathsf{Rel}$$

The functor $F : \mathsf{Set} \to \mathsf{Rel}$ behaves as the identity on objects, i.e., $FX = X$, and for a function $f : X \to Y$ in Set, $Ff : FX \to FY$ is a graph $Ff \subseteq X \times Y$ defined as $\{(x, f(x)) \ : \ x \in X\}$. On the other direction, $U : \mathsf{Rel} \to \mathsf{Set}$ is given by the powerset functor, $\mathcal{P}X$. We claim that it is the same to give a relation from FX to Y than a function from X to UY, this is, $\mathsf{Rel}(FX, Y) \cong \mathsf{Set}(X, UY)$, i.e., giving a relation from $X \to Y$ is no different from having a set from $X \to \mathcal{P}Y$.

Exercise 12. Show that $F : \mathsf{Set} \to \mathsf{Rel}$ is a functor (this is, verify the functor axioms).

Example 29. Using above's adjunction, note that $\mathsf{Rel}(1, 1) \cong \mathsf{Set}(1, 2)$. This looks similar to something we have seen before. We claim that if we fix a language L over an alphabet A, and

consider the set of automata on $\mathsf{Auto}(L)$ on $\mathsf{Set}(1,2)$, the deterministic automata on L, and $\mathsf{Auto}(L)$ on $\mathsf{Rel}(1,1)$, the nondeterministic automata, we get the adjunction

$$\mathsf{DA}(L) \underset{U}{\overset{F}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} \mathsf{NDA}(L)$$

where U is the powerset construction from theory of computation.

We have the following important theorem:

Theorem 2. Left adjoints preserve colimits, right adjoints preserve limits.

The proof is omitted here, but we leave the following special case as an exercise:

Exercise 13. Right adjoints preserve products.

Naturality Where does the notion of naturality come from in the definition of an adjunction? As noted in the limit discussion, it always comes from a natural isomorphism between functors. For adjunctions, there are two functors at play, one giving the naturality in X while the other gives naturality in Y. Naturality in X comes from the natural isomorphism between the contravariant functors $\mathscr{D}(F-,Y)$ and $\mathscr{C}(-,GY)$. For a treatment on the contravariant case, see the discussion above on limits. The other natural isomorphism, providing naturality in Y, arises between covariant functors $\mathscr{D}(FX,-)$ and $\mathscr{C}(X,G-)$[2] which we describe as follows:

We first explicitly define the functors $\mathscr{D}(FX,-):\mathscr{D}\to\mathsf{Set}$ and $\mathscr{C}(X,G-):\mathscr{D}\to\mathsf{Set}$.
On objects:

$$Y \overset{\mathscr{D}(FX,-)}{\longmapsto} \mathscr{D}(FX,Y) \quad \text{and} \quad Y \overset{\mathscr{C}(X,G-)}{\longmapsto} \mathscr{C}(X,GY)$$

On morphisms:

$$(g: Y\to Z) \overset{\mathscr{D}(FX,-)}{\longmapsto} (\mathscr{D}(FX,Y) \xrightarrow{\mathscr{D}(FX,g)} \mathscr{D}(FX,Z))$$

and

$$(g: Y\to Z) \overset{\mathscr{C}(X,G-)}{\longmapsto} (\mathscr{C}(X,GY) \xrightarrow{\mathscr{C}(X,Gg)} \mathscr{C}(X,GZ)),$$

where

$$(FX \xrightarrow{v} Y) \overset{\mathscr{D}(FX,g)}{\longmapsto} (FX \xrightarrow{v} Y \xrightarrow{g} Z) \quad \text{and} \quad (X \xrightarrow{u} GY) \overset{\mathscr{C}(X,Gg)}{\longmapsto} (X \xrightarrow{u} GY \xrightarrow{Gg} GZ).$$

Finally,

$$1_Y \overset{\mathscr{D}(FX,-)}{\longmapsto} 1_{\mathscr{D}(FX,Y)} \quad \text{and} \quad 1_Y \overset{\mathscr{C}(X,G-)}{\longmapsto} 1_{\mathscr{C}(X,GY)}.$$

It is left as a quick exercise to check that these functors preserve composition.

Now we can continue. We have the natural isomorphism

$$\eta : \mathscr{D}(FX,-) \Rightarrow \mathscr{C}(X,G-).$$

[2]These four functors are all examples of *representable functors*.

Expanded, we have a family of isomorphisms $(\eta_Y : \mathscr{D}(FX, Y) \to \mathscr{C}(X, GY))_{Y \in \mathrm{ob}(\mathscr{D})}$, such that for every morphism $g : Y \to Z$, the following diagram commutes:

$$\begin{array}{ccc} \mathscr{D}(FX,Y) & \xrightarrow{\eta_Y} & \mathscr{C}(X,GY) \\ {\scriptstyle \mathscr{D}(FX,f)}\downarrow & & \downarrow{\scriptstyle \mathscr{C}(X,Gf)} \\ \mathscr{D}(FX,Z) & \xrightarrow[\eta_Z]{} & \mathscr{C}(X,GZ) \end{array}$$

Notice that this exactly describes the identity

$$\eta_Z(g \circ v) = Gg \circ \eta_Y(v)$$

for every $v : FX \to Y$. Now if we call η by $(-)_\sharp$ instead, we exactly replicate (3):

$$(g \circ v)_\sharp = Gg \circ v_\sharp.$$

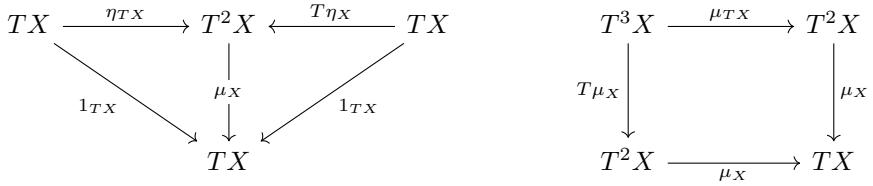
6 Monads and Algebras

Ever since the work of Moggi [3], monads have been ubiquitous in computer science. To motivate the topic, we will begin by looking at the algebra for a monad. An *algebraic theory* is given by a signature Σ, which is a set of operations of a given arity, and a set of equations E. Given our signature Σ, a monad T_Σ maps a set X to the set of Σ-terms where each term has $x \in X$ as variables, and a morphism $f : X \to Y$ to a term over Y by substituting the elements $x \in X$ according to f. The unit of this monad T_Σ is given by the identity (i.e., it maps each element of X to itself) and the multiplication by term composition. Expanding this monad with E, our set of equations, we can form now a monad $T_{\Sigma,E}$, which quotients the set of Σ-terms by E. One way of looking at monads is thus by thinking about them as a consistent way for forming expressions over a fixed set, together with a set of operations that can be used on these.

Definition 33 (Monads). Let $\mathscr{C}$ be a category. A monad on $\mathscr{C}$ is a triple (T, μ, η) where:

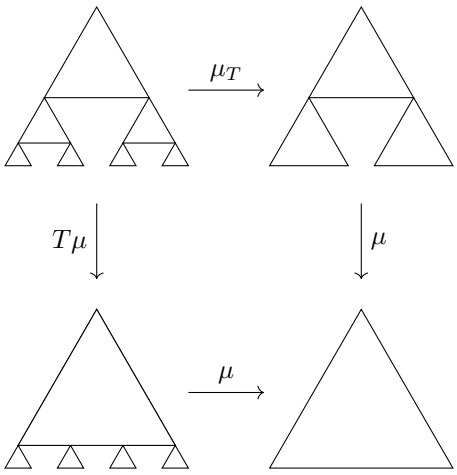
- $T : \mathscr{C} \to \mathscr{C}$ a functor
- $\mu : T^2 \Rightarrow T$ a natural transformation called "multiplication"
- $\eta : \mathbb{1}_{\mathscr{C}} \Rightarrow T$ a natural transformation called "unit"

Such that for each X, the following diagrams commute:



The two commuting triangles on the left specify how the "unit" and "multiplication" maps interact. The commuting square on the right means that the "multiplication" map is associative in some sense.

Note 3. We can see what the above commuting square does by intuitively looking at a collection of terms of terms of terms. There are two ways in which we can flatten these. Either we take a bottom-up approach, or we start from the top. These two ways of flattening our terms correspond to the maps $T\mu_X$ and μ_{TX} respectively. Finally, we can flatten our terms once more by using the monad multiplication, thus ending with a collection where all our original terms are contained. Pictorially:



Whenever we have a category of alebras, we can come up with a monad. Let us look at Mon, the category of monoids. We will associate it with the following monad on $T : \mathsf{Set} \to \mathsf{Set}$. A set X is mapped to X^*, the set of finite strings over X (including the empty one, which is the unit of our monoid),

$$X \mapsto X^* = \{\}$$

In order to give a monad, we have to provide a natural transformation from the identity on Set to T, $\eta : 1_{\mathsf{Set}} \Rightarrow T$, we do this by mapping an element $x \in X$ to a singleton list containing only x:

$$\begin{aligned} \eta_X : X &\to TX \\ X &\to X^* \\ x &\mapsto [x] \end{aligned}$$

The last piece of data that we have to give to have a monad is the multiplication, which will go from lists of lists of X to lists of X,

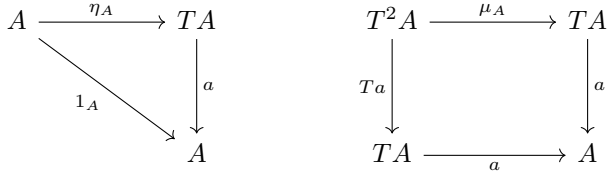
$$\begin{aligned} \mu_X : T^2X &\to TX \\ \mu_X : (X^*)^* &\to X^* \\ [[x_{11}, \ldots, x_{1n}], [x_{21}, \ldots, x_{2n}], \ldots, [x_{m1}, \ldots, x_{mn}]] &\mapsto [x_{11}, \ldots, x_{mn}] \end{aligned}$$

Notice that this is just concatenation. Let us spell out this:

Example 30 (List monad). Let us consider the following functor in Set, usually called the *list functor*, $\mathcal{L} : \mathsf{Set} \to \mathsf{Set}$. It acts on a set X by forming a list $\mathcal{L}X$ whose elements are exactly the elements $x \in X$, i.e., $[x_0, x_1, \ldots, x_n]$, and given a function $f : X \to Y$, it induces a function $\mathcal{L}f : \mathcal{L}X \to \mathcal{L}Y$ by applying f elementwise to members of the list $\mathcal{L}X$. We can equip this functor with a monad structure by considering the unit $\eta^{\mathcal{L}}$ to be the list with a single element, $[x]$, and the multiplication $\mu^{\mathcal{L}} : \mathcal{L}^2X \to \mathcal{L}X$ to be given by flattening a double list.

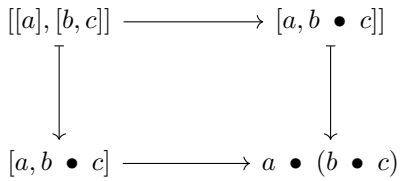
This is all good, but where are all the monoids? How can we recover from the above monad the category of all monoids, Mon? Let us assume we have a set A and we put a monoid structure on A, this is, we must give a way of multiplying elements of A. This corresponds to giving a map $A^* \xrightarrow{a} A$. We introduce now the following:

Definition 34 (Algebras over a monad). Let (T, μ, η) be a monad on category $\mathscr{C}$. An algebra for $\mathscr{C}$ is a morphism $a : TA \to A$ in $\mathscr{C}$, where A is an object of $\mathscr{C}$, and the following diagrams commute:



The above not only gives us a way for multiplying terms of A (through the map η_A), but a way to flatten elements of A^* to end up with elements in A ($a : TA \to A$).

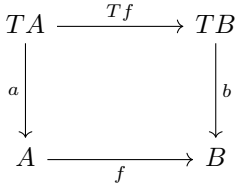
Let us look at a concrete example. If the algebra that we consider is such that TX is X^*, the above square acting on a list of elements in X, $[a, b, c]$,



We can now see that the above diagrams are telling us that that X^* is nothing else but a monoid.

Proposition 1. An algebra for the free monoid functor T as defined earlier is just a monoid.

Definition 35 (Algebra homomorphisms). If $a : TA \to A$ and $b : TB \to B$ are algebras for T. A morphism from a to b is defined to be a morphism $f : A \to B$ in $\mathscr{C}$ such that



Exercise 14. Convince yourself that the above with $TX = X^*$ gives us precisely the monoid homomorphisms.

Definition 36 (Eilenberg-Moore Algebras). The Eibenberg-Moore category over a monad (T, μ, η) is denoted $\mathbf{EM}(T)$. It is also called the category of algebras over the monad T. It has objects the algebras over T, and morphisms the algebra homomorphisms.

We can now examine a subcategory of these algebras, the Kleisli category of a monad T, which is equivalent to the category of free $T-$algebras. These will be determined by its set of generators. We start by defining what a Kleisli category is,

Definition 37 (Kleisli Category). Let T be a monad over $\mathscr{C}$. The Kleisli category $\mathcal{K}l(T)$ over T has:

- Objects exactly the objects of $\mathscr{C}$

- Morphisms are morphisms $f : A \to TB$ in $\mathscr{C}$

For each object $X \in \mathcal{K}l(T)$, the identity morphisms of X in $\mathcal{K}l(T)$, which is a morphism $X \to TX$ in $\mathscr{C}$, is just the component of the monad unit η_X. The composition of two morphisms $g : B \to C, f : A \to B$ in $\mathcal{K}l(T)$, which are morphisms $g : B \to TC$ and $f : A \to TB$ in $\mathscr{C}$, is $g \circ f \triangleq \mu_C \circ (Tg) \circ f$, where the compositions on the right hand side happen in $\mathscr{C}$.

$$A \xrightarrow{\quad f \quad} TB \xrightarrow{\quad Tg \quad} TTC \xrightarrow{\quad \mu \quad} TC$$

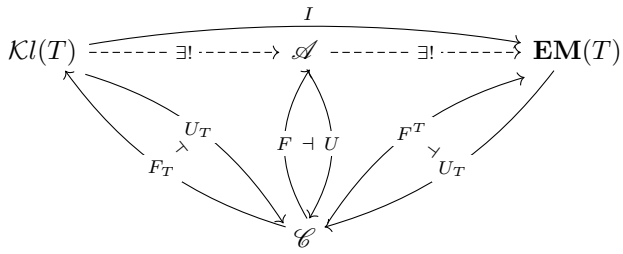
As stated above, a free algebra is determined by its set of generators. If we want to give a morphism from $A^* \to B^*$ (i.e., from the free algebra on A to the free algebra on B) , it suffices to tell what this morphism is doing on the generators, i.e., it is enough to have a morphism $A \to B^*$.

Then if we are interested in a free algebra, we can keep the elements of our category $\mathcal{C}$ and consider the arrows $A \to TB$. This is, we take the Kleisli category of T

Note 4. The objects in the Kleisli category are the "free algebras". For example, if T is the free group monad, then $\mathcal{K}l(T)$ is the full subcategory of $\mathsf{Grp} \simeq \mathbf{EM}(T)$ where the objects are all the free groups. In general, $\mathcal{K}l(T)$ can be viewed as a full subcategory of $\mathbf{EM}(T)$ with only the free objects.

Exercise 15. Prove $\mathcal{K}l(T)$ is actually a category, i.e. that the category axioms for associatity of composition and the equations for identity morphisms actually hold.

If we start with a monad $T : \mathcal{C} \to \mathcal{C}$ and take an adjunction $F_T \dashv U_T$, then we can get the monad back by grabbing the composite of the two functors, i.e., $T = U_T \circ F_T$. In general, we may have different adjunctions giving us the monad T, but there are two very special ones. Namely, the $\mathcal{K}l(T)$ and $\mathbf{EM}(T)$. If we start with an adjunction $F_T \dashv U_T$ we can construct $\mathcal{K}l(T)$, and similarly, $\mathbf{EM}(T)$ can be constructed from $F^T \dashv U^T$. Diagrammatically,



Proposition 2. $\mathcal{K}l(T)$ and $\mathbf{EM}(T)$ are initial and terminal among adjunctions.

Examples of monads are listed below.

Definition 38 (Finitary distribution monad). Let $\mathcal{D} : \mathsf{Set} \to \mathsf{Set}$ be the finitary distribution functor, which sends a set X to the set $\mathcal{D}X$ of all finitely supported probability distributions onf X,

$$\mathcal{D}X = \{\varphi : X \to [0,1] : \sum_{x \in X} \varphi(x) = 1, \mathsf{supp}(\varphi) \text{ is finite}\}$$

and a funtion $f : X \to Y$ to $\mathcal{D}f : \mathcal{D}X \to \mathcal{D}Y$, the pushforward of measures along f, this is,

$$\mathcal{D}f(\varphi) = \lambda y. \sum_{x \in f^{-1}(y)} \varphi(x).$$

The finitary distribution monad $(\mathcal{D}, \eta^{\mathcal{D}}, \mu^{\mathcal{D}})$ is given by taking the unit $\eta^{\mathcal{D}} : X \to \mathcal{D}X$ to be the Dirac distribution, $\eta^{\mathcal{D}}(x) = \delta_X$ and we define the multiplication $\mu^{\mathcal{D}} : \mathcal{D}^2X \to \mathcal{D}X$ by

$$\mu^{\mathcal{D}}(\Phi)(x) = \sum_{\varphi \in \mathcal{D}X} \Phi(\varphi)\varphi(x).$$

Note 5. The distribution monad captures probabilistic behavior.

- $Kl(\mathcal{D})$ is $\mathsf{FinStoch}$, the category of finitary stochastic maps.
- $EM(\mathcal{D})$ is the category of convex algebras.

Definition 39 (Powerset monad). Let $\mathcal{P} : \mathsf{Set} \to \mathsf{Set}$ be the powerset functor, which sends a set X to its powerset $\mathcal{P}X$ and maps functions $f : X \to Y$ to $\mathcal{P}f : \mathcal{P}X \to \mathcal{P}Y$, which sends sets to their image under f. This is, $(\mathcal{P}f)(X) := f[X] = \{y \in Y : y = f(x) \text{ for some } x \in X\}$. The powerset monad $(\mathcal{P}, \eta^{\mathcal{P}}, \mu^{\mathcal{P}})$ is given by taking the unit $\eta^{\mathcal{P}} : X \to \mathcal{P}X$ to be the singleton set, $\eta^{\mathcal{P}}(x) = \{x\}$ and the multiplication $\mu^{\mathcal{P}} : \mathcal{P}^2X \to \mathcal{P}X$ the common union of sets,

$$\mu^{\mathcal{P}}(\{X_i \in \mathcal{P}X : i \in I\}) = \bigcup_{i \in I} X_i.$$

Note 6. The powerset monad captures nondeterminism.

- $\mathcal{K}l(\mathcal{P})$ is Rel, the category of sets and binary relations.
- $\mathbf{EM}(\mathcal{P})$ is the category of complete join semilattices.

References

[1] S. Awodey. *Category Theory*. Oxford Logic Guides. OUP Oxford, 2010. ISBN: 9780191612558.

[2] Tom Leinster. *Basic Category Theory*. 2016. arXiv: 1612.09375 [math.CT].

[3] Eugenio Moggi. "Notions of computation and monads". In: *Information and Computation* 93.1 (1991). Selections from 1989 IEEE Symposium on Logic in Computer Science, pp. 55–92. ISSN: 0890-5401. DOI: https://doi.org/10.1016/0890-5401(91)90052-4.