Category Theory 1

Yuexin Liao

November 2023

1 Introduction

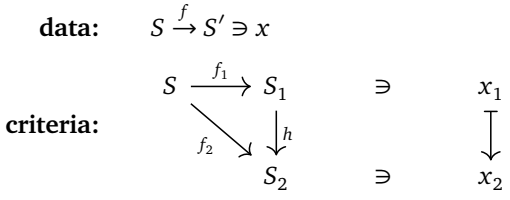
In delving deeper into the exposition of category theory, we aim to establish a thorough understanding that bridges the gap between concrete examples and abstract concepts. At the heart of category theory lie universal properties, encompassing notions of "initial" and "final." Instead of immediately providing a formal definition of a category, our approach initially involves exploring these "universal properties" through illustrative problems and intuitive language. These universal properties serve as valuable tools for gaining a profound insight into the fundamental logic underpinning computer science and data science. I will write several essays to clarify that and this is the first essay.

2 Mathematical Construction

We will begin with known mathematical constructions, viewed from a universal, categorical point of view.

2.1 Adding an element to a set

Given a set S, consider



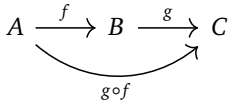
The function $h : S_1 \to S_2$ must satisfy $h(x_1) = x_2$ and $\forall x \in S.\, h(f_1(x)) = f_2(x)$.

Commutative Diagrams

Diagrams such as the above implicitly assume that compositions commute (that's why they're called commutative diagrams). We can express the above more concisely as:

$$h \circ f_1 = f_2 \quad \text{or} \quad h f_1 = f_2$$

where



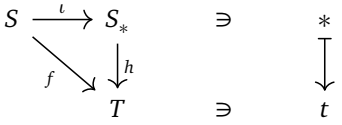
with $(g \circ f)(a) \stackrel{\text{def}}{=} g(f(a))$ for all $a \in A$.

Initial (Universal) Solution

An initial solution is one that uniquely maps to any other solution. Note that such a solution doesn't always exist. In the case of adding a single element to a set, the problem can be specified in terms of finding an appropriate $S \xrightarrow{f} T \ni t$ such that

$$\forall S \xrightarrow{f'} S' \ni x \;.\; \exists!\; h : T \to S' \text{ such that } h(t) = x \text{ and } h \circ f = f'$$

We present our solution:



where S_* is defined as $\{*\} \cup \{[x] \mid x \in S\}$ and $\iota : S \to S_*$ as $\iota(x) = [x]$ for all $x \in S$. (The set S_* is an *implementation* of the disjoint union of $\{*\}$ and S; the square brackets tag the elements from S.)

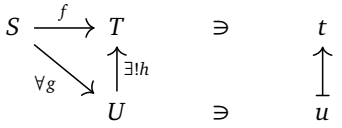
Define h, for $z \in S_*$, as

$$h(z) = \begin{cases} t & \text{if } z = * \\ f(x) & \text{if } z = [x] \end{cases}$$

For universality we must also prove that this is a unique solution. Consider $k : S_* \to T$ that satisfies $k \circ \iota = f$ and $k(*) = t$. Then, we are required to prove (RTP): $k(z) = h(z)\ \forall z \in S_*$.

Final solution

There is a second kind of universal property, callled a final solution: any other solution admits a unique map to the final one, again examining the case of adding an element to a set, our criteria now is inverted. As we are looking for a final solution, any $U \ni u$ must admit a unique map to our solution $T \ni t$, as shown below.



For a set A, let $!_A$ be the unique function $A \to \{*\}$ given by the constantly $*$ function, that is, $!_A(a) = *$ for all $a \in A$.

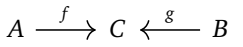
Then, $T = \{*\}$, $t = *$, and $f = !_S$ is a final solution.

(Note the correspondence between $* \in \{*\}$ and `():unit` in most functional programming languages.)

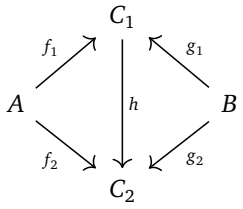
2.2 Adding two sets

Given A and B sets.

data:



criteria:



That is, such that (1) h is a function $C_1 \to C_2$ and (2) it makes the diagram commute (*i.e.* $h \circ f_1 = f_2$ and $h \circ g_1 = g_2$).

Exercise 1. Find the initial solution.

Note We will illustrate the core logic of how to solve it in Section 3.

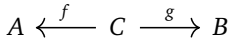
2.3 Duality

When you have a problem in "one direction" you can turn it around and get a problem in "the other direction". This involves flipping the directions of the arrows in the commutative diagrams.

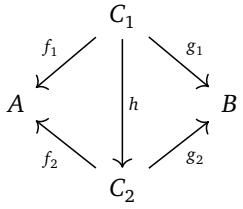
Example: Dual form of adding two sets

Given two sets A and B.

data:



criteria:



That is, such that (1) h is a function $C_1 \to C_2$ and (2) it makes the diagram commute (*i.e.* $h \circ f_2 = f_1$ and $h \circ g_2 = g_1$).

Exercise 2. Find the final solution.

2.4 Generating a monoid

Background on Monoids

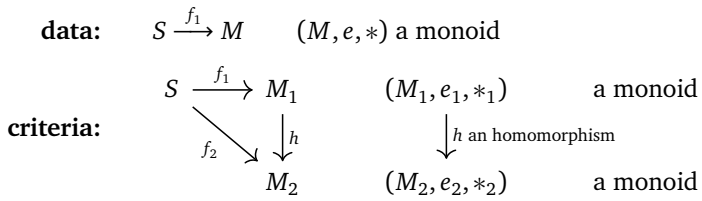
A *monoid* is a triple $(M, e, *)$ where M is a set, $e \in M$ is a neutral element, and $*$ is a binary operation on M; where e and $*$ obey:

$$\forall x \in M \,.\, e * x = x \text{ and } x * e = x \qquad \text{(neutral element)}$$
$$\forall x, y, z \in M \,.\, (x * y) * z = x * (y * z) \qquad \text{(associativity)}$$

Monoid homomorphism

A monoid homomorphism is a function between (the underlying sets of) two monoids that preserves the unit and the multiplication. Formally, $h : (M_1, e_1, *_1) \to (M_2, e_2, *_2)$ is a function $h : M_1 \to M_2$ such that $h(e_1) = e_2$ and $\forall x, y \in M_1 \,.\, h(x *_1 y) = h(x) *_2 h(y)$.

Our next exercise will deal with the *free monoid* on a set. Given a set S, we would like to construct a monoid. In the language we've been using, we can present it as follows:



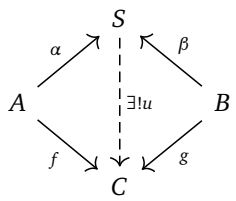
Exercise 3. Find the initial solution in this case.

Note We will give specific hints in section 3.

3 Hints To Exercises

3.1 Adding two sets

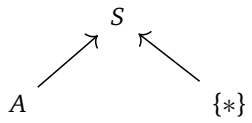
Recall that in Exercise 1, we try to look for a universal initial solution to adding sets A and B:



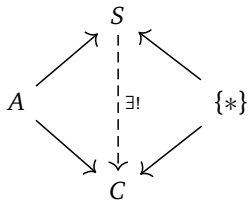
for all sets C, and functions f and g.

This is a good strategy that when dealing with a problem we do not know immediately how to solve, it might be useful to somehow 'simplify' it in order to get some insight. Nevertheless, it is not so much that we lose all structural information.

Consider a special instance of this problem, where B is a singleton set $\{*\}$:



Note that $\{*\} \to S$ is mapping $*$ to some element in S, and overall this represents the problem of adding an element to A (Subsection 2.1). We want to make sure that there is a unique mapping to any other solution C, as illustrated below.



Here S has to preserve the structure of A and be augmented with an additional element in some way. The image of elements of A in S have to be distinct from the image of $*$, since otherwise there would not be a unique way of representing the two in C. This is a map $a \mapsto [a]$ for $a \in A$. Compare this to the previous problem, where we considered $A \to A_* = \{*\} \cup \{[x] \mid x \in A\}$.

Taking this 'tagging' idea further and generalising, we may consider the set:

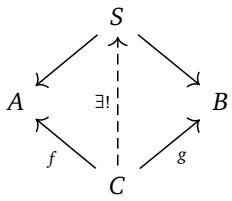
$$A \uplus B = \{(0, a) \mid a \in A\} \cup \{(1, b) \mid b \in B\}$$

We claim that S should be $A \uplus B$, and that $\alpha : a \mapsto (0, a)$, $\beta : b \mapsto (1, b)$ should be the maps for the initial solution.

It can lead to a proposition that this is indeed an initial solution.

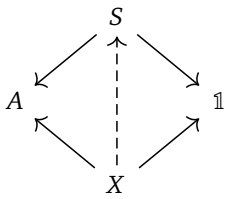
3.2 The dual problem

In Exercise 2, we try to find the dual solution to that described previously:

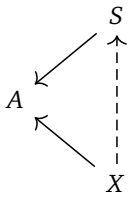


This is a more involved problem than adding two sets, and simplifications might not immediately reveal what needs to be done. Even so, to give hints, we will test different structures for B to gain some intuition.

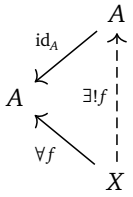
First, consider B to be a singleton set, $B = \mathbb{1}$, where one typically writes $\mathbb{1}$ for a generic singleton set. Then the problem becomes:



There is only one function $X \to \mathbb{1}$ and only one function $S \to \mathbb{1}$ (they both map everything to the only element of $\mathbb{1}$), so the problem is reduced to:

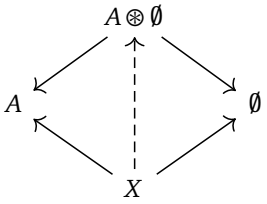


and the final solution to this is clear:



In general let's say S is of the form $A \circledast B$ for some construction $\circledast$, where for $B = \mathbb{1}$ we can take $A \circledast B$ to be A (*i.e.* the structure matches that of of A).

Then we consider the case that if $B = \emptyset$



This is only possible and valid when $A \circledast \emptyset = \emptyset$ because there are no functions from a nonempty set into an empty set. Also recall that there is only one function from $\emptyset$ to any other set – namely the empty function (the function with the empty graph).

Now, we can consider another case when $B = \{0, 1\} =$ — In another words, what should $A \circledast$ be? This extension gives us some insight into the general solution.

4 Isomorphism of initial solutions

Recall that in our initial solution to the problem of generating a monoid from a set, we actually found a number of different solutions, all of which seemed somehow structurally similar:

1. $S \to \text{List}(S)$, where $\text{List}(S)$ is the monoid of lists whose items have values drawn from S, with the empty list $[\,]$ being the neutral element and the append operator $@$ being the binary operation.

2. $S \to S^*$, where S^* is the set of all strings/words on S. This is a monoid if the neutral element is ϵ (the empty word) and the operation is string concatenation $\cdot$.

3. For $n \in \mathbb{N}$ define S_n in the following way:

$$S_0 = \{(0, ())\}$$

and for $n > 0$:

$$S_n = \{(n, (s_1, \ldots, s_n)) \mid s_1, \ldots, s_n \in S\}$$

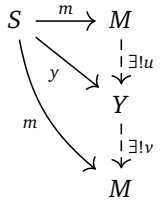
Then $\bigcup_{n\in\mathbb{N}} S_n$ is a monoid with the operation $\cdot$ defined as:

$$(l, s_1, \ldots, s_l) \cdot (k, (s'_1, \ldots, s'_k)) = (l + k, (s_1, \ldots, s_l, s'_1, \ldots, s'_k))$$

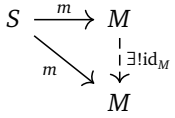
and with $(0, ())$ being the neutral element.

Each of these solutions are indeed 'essentially the same' and this is a fundamental property of universal solutions. This concept (isomorphism) will be defined properly once we start dealing with category theory, but for now we can still prove that two initial solutions of the kind we've been dealing with are isomorphic, *i.e.* that they are in structural bijective correspondence.

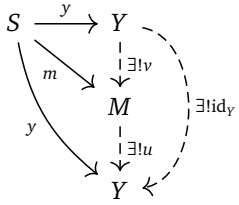
Given two initial solutions M (Mine) and Y (Yours), diagrammatically:



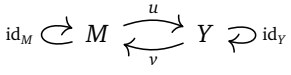
Note that we have also:



Thus $v \circ u = \mathrm{id}_M$. Analogously we conclude that $u \circ v = \mathrm{id}_Y$ because:



From this we conclude that *any two initial solutions are isormorphic*:



From the perspective of computer science, we may think of universal problems as implementation i.e. independent specifications for which concrete solutions provide actual implementations, all of which have to necessarily be essentially the same — structure should be preserved regardless of the underlying implementation (it doesn't matter whether we're dealing with strings, lists, *etc.*).

Exercise 4. From a set S generate a *commutative* monoid M, *i.e.* one that satisfies $x \cdot y = y \cdot x$ for all $x, y \in M$ — find the initial solution.

Solution. The intuition behind the solution is that the free commutative monoid on a set is given by its *set of finite multisets*. Here is a formal implementation of this idea.

Let $\approx$ be the binary relation on $S^* \stackrel{\text{def}}{=} \bigcup_{\ell \in \mathbb{N}} \{\ell\} \times S^\ell$ given by

$$\begin{aligned} & \big(i, (s_1, \ldots, s_i)\big) \approx \big(j, (t_1, \ldots, t_j)\big) \\ \iff & \\ & \text{there exists a bijection } \sigma : \{1, \ldots, i\} \to \{1, \ldots, j\} \text{ such that } s_k = t_{\sigma(k)} \text{ for all } k = 1, \ldots, i \ . \end{aligned}$$

• Show that $\approx$ is an equivalence relation.

Define the set

$$\mathcal{M}_f(S) \stackrel{\text{def}}{=} S^*/_{\approx} \ .$$

• Show that

$$\begin{aligned} & \big(i, (s_1, \ldots, s_i)\big) \approx \big(i', (s'_1, \ldots, s'_{i'})\big) \text{ and } \big(j, (t_1, \ldots, t_j)\big) \approx \big(j', (t'_1, \ldots, t'_{j'})\big) \\ & \text{imply } \big(i + j, (s_1, \ldots, s_i, t_1, \ldots, t_j)\big) \approx \big(i' + j', (s'_1, \ldots, s'_{i'}, t'_1, \ldots, t'_{j'})\big) \ . \end{aligned}$$

We therefore have a binary operation $\oplus : \mathcal{M}_f(S) \times \mathcal{M}_f(S) \to \mathcal{M}_f(S)$ given by

$$\big[\big(i, (s_1, \ldots, s_i)\big)\big]_{\approx} \oplus \big[\big(j, (t_1, \ldots, t_j)\big)\big]_{\approx} \stackrel{\text{def}}{=} \big[\big(i + j, (s_1, \ldots, s_i, t_1, \ldots, t_j)\big)\big]_{\approx}$$

• Prove that

$$\big(\ \mathcal{M}_f(S) \ , \ \big\{\big(0, ()\big)\big\} \ , \ \oplus \ \big)$$

is a commutative monoid.

We claim that the function

$$\iota : S \to \mathcal{M}_f(S) : s \mapsto \big\{\big(1, (s)\big)\big\}$$

is an initial universal solution.

To this end, consider a commutative monoid $(M, \text{ı}, *)$ and a function $f : S \to M$.

• Prove that

$$\big(i, (s_1, \ldots, s_i)\big) \approx \big(i', (s'_1, \ldots, s'_{i'})\big) \implies *_i\big(f(s_1), \ldots, f(s_i)\big) = *_{i'}\big(f(s_1), \ldots, f(s'_{i'})\big)$$

where $*_0() \stackrel{\text{def}}{=} \text{ı}$ and, for $n \in \mathbb{N}$, $*_{n+1}(x_1, \ldots, x_n, x_{n+1}) \stackrel{\text{def}}{=} *_n(x_1, \ldots, x_n) * x_{n+1}$.

We therefore have a function $f^\# : \mathcal{M}_f(S) \to M$ given by

$$f^\#\big(i, (s_1, \ldots, s_i)\big) \stackrel{\text{def}}{=} *_i\big(f(s_1), \ldots, f(s_i)\big) \ .$$

• Prove that $f^\#$ is a monoid homomorphism.

Assume that $h : \mathcal{M}_f(S) \to M$ is a monoid homomorphism such that $h \circ \iota = f$.

• Prove that $h = f^\#$. □

Exercise 5. From a set S generate a commutative and *idempotent* monoid M *i.e.* one that also satisfies $x \cdot x = x$ for all $x \in M$ — find the initial solution.

5 Orders and lattices

For a set P and $\leq \subseteq P \times P$ a binary relation, we will consider the following properties.

Definition 1. Reflexivity, transitivity, antisymmetry

- $\leq$ is *reflexive* if $\forall a \in P. \quad a \leq a$
- $\leq$ is *transitive* if $\forall a, b, c \in P. \quad a \leq b \wedge b \leq c \Rightarrow a \leq c$
- $\leq$ is *antisymmetric* if $\forall a, b \in P. \quad a \leq b \wedge b \leq a \Rightarrow a = b$

There is standard terminology for binary relations subject to such axioms.

Definition 2. Partial orders and preorders

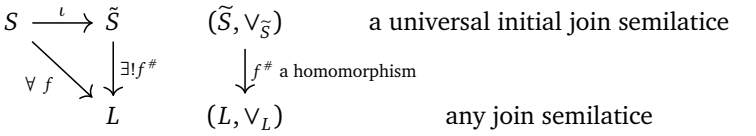
- $(P, \leq)$ is a *preorder* (or *quasiorder*) if $\leq$ is reflexive and transitive
- $(P, \leq)$ is a *partial order* if is a preorder ($\leq$ is reflexive and transitive) and $\leq$ is also antisymmetric

Definition 3. A partial order $(P, \leq)$ equipped with an associative, commutative, and idempotent binary operation $\vee$, typically referred to as *join* (or *sup*), that satisfies $x \leq y \iff x \vee y = x$ is a *join (or sup) semilattice*.

(It may also have a least (or bottom) element $\bot \in L$ satisfying $\forall x \in L, \ \bot \vee x = \bot$.)

Example

We can now consider the problem of generating a join semilattice. For a set S, we want to freely construct a join semilatice $(\widetilde{S}, \vee_{\widetilde{S}})$ as follows:



where h must preserve structure, *i.e.* $f^{\#}(x \vee_{\tilde{S}} y) = f^{\#}(x) \vee_L f^{\#}(y)$

The exercise is to find the initial solution here, but we will start with some intuition. The join semilatice generated from S must contain each element $x \in S$. Similarly, for any two, three, or, more generally, any finite number of distinct elements of S, it must contain the join or union of all of them. Since the operator is commutative, the order doesn't matter, and since it's idempotent, only one copy of each element of S can exist. The picture looks something like this:

$$
\begin{array}{lrcl}
 & x & \leftrightarrow & \{x\} \subseteq S \\
x \neq y & x \vee y & \leftrightarrow & \{x, y\} \subseteq S \\
x \neq y \neq z \neq x & x \vee y \vee z & \leftrightarrow & \{x, y, z\} \subseteq S \\
 & & \vdots & \\
x_i \neq x_j \ \forall i, j & x_1 \vee \cdots \vee x_n & \leftrightarrow & \{x_1, \ldots, x_n\} \subseteq S
\end{array}
$$

Thus, we might guess that the universal initial solution $\widetilde{S}$ is the collection of finite subsets of S: $\mathcal{P}_{\text{fin}}(S) = \{X \mid X \subseteq_{\text{fin}} S\}$, with the join operator $X \vee Y = X \cup Y$. This is close, but not entirely correct.

Exercise 6. There is a 'bug' in the above proposed solution. Find the actual initial solution.

Definitions

Definition 4. Categories (internal viewpoint)

A category consists of two kinds of entities:

- Objects
- Morphisms between objects (also called maps or arrows)

Every morphism comes with an assigned domain (also source) and codomain (also target). We write $D \xrightarrow{f} C$ to show that the morphism f has domain D and codomain C.

Every object A has an associated *identity* arrow $A \xrightarrow{\mathrm{id}_A} A$, also denoted 1_A or just A.

Arrows come with a (well-typed) *composition* operation $\circ$. Given $A \xrightarrow{f} B$ and $B \xrightarrow{g} C$ we have $A \xrightarrow{g \circ f} C$, also denoted $A \xrightarrow{gf} B$ or $A \xrightarrow{f;g} B$ (read "apply f and then g").

Composition and identity arrows must satisfy laws similar to the monoid laws:

- Composition is associative. Given any objects and arrows as follows

  $$A \xrightarrow{h} B \xrightarrow{g} C \xrightarrow{f} D$$

  we require that

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

- id_X is an identity under composition. Given any

  $$X \xrightarrow{k} Y$$

  we require that

  $$k \circ \mathrm{id}_X = k = \mathrm{id}_Y \circ k.$$

Definition 5. Categories (enriched viewpoint)

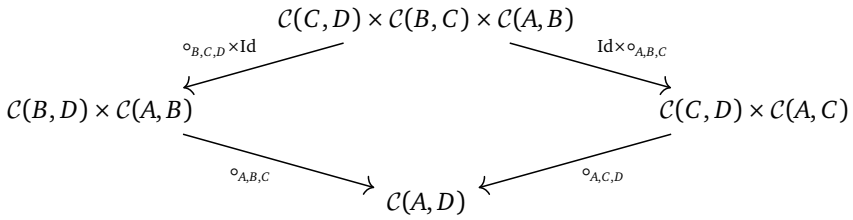
A category $\mathcal{C}$ consists of

- a class of objects $Obj(\mathcal{C})$ (also written $|\mathcal{C}|$), together with
- *hom-sets* $\mathcal{C}(A, B)$ for all $A, B \in Obj(\mathcal{C})$, and
- identities $\mathrm{id}_A \in \mathcal{C}(A, A)$ for all $A \in Obj(\mathcal{C})$, in addition to
- a composition operation

$$\begin{array}{ccccccc} \mathcal{C}(B,C) & \times & \mathcal{C}(A,B) & \xrightarrow{\circ_{A,B,C}} & \mathcal{C}(A,C) \\ g & , & f & \mapsto & g \circ f \end{array}$$

It must satisfy laws, written here as commutative diagrams:

- Composition is associative:



Here, $\mathrm{Id} \stackrel{\text{def}}{=} (x \mapsto x)$ is the standard identity function. The operator $\times$ on two functions intuitively runs two them in parallel. Given

$$f : A_1 \to B_1 \ , \quad g : A_2 \to B_2$$

we have

$$f \times g : A_1 \times A_2 \to B_1 \times B_2$$

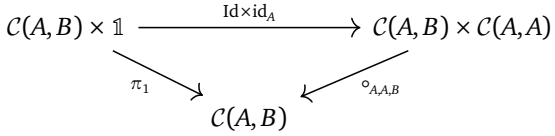
defined by

$$\big(f \times g\big)(x, y) \mapsto (f\,x, g\,y)$$

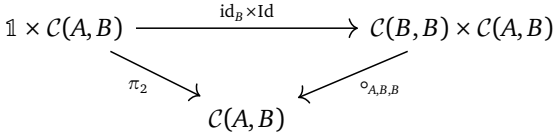
- Identity is a left and right neutral element. First, notice that we can write

$$\mathbb{1} \xrightarrow{\mathrm{id}_A} \mathcal{C}(A,A)$$

to express the idea that the identity morphism id_A exists for every A. Then the right identity law is expressed as:



and the left identity law is expressed similarly as:



6 Examples of categories

6.1 Sets

Set is a category whose objects are sets and whose morphisms are standard set-theoretic functions. The identity is the standard identity function and composition is the standard composition of functions. The set of morphisms can be written as

$$\mathbf{Set}(S,T) = \{f \subseteq S \times T \mid f \text{ is a function}\}$$

6.2 Monoids

Mon is a category whose objects are monoids $(M, \mathsf{I}, \star)$ and whose morphisms are monoid homomorphisms (*i.e.* functions which preserve the monoid structure).

Identity arrows are the usual identity functions and composition is the usual function composition.

We can check that composition of monoid homomorphisms always produces a monoid homomorphism. Assume we have monoids

$$(M_1, \mathsf{I}_1, \star_1) \ , \quad (M_2, \mathsf{I}_2, \star_2) \ , \quad (M_3, \mathsf{I}_3, \star_3) \ .$$

and let $f : M_1 \to M_2$ and $g : M_2 \to M_3$ be monoid homomorphisms.

Then,

$$\big(g \circ f\big)(\mathsf{I}_1) \ = \ g\big(f(\mathsf{I}_1)\big) \ = \ g(\mathsf{I}_2) \ = \mathsf{I}_3$$

and, for $x, y \in M_1$,

$$\begin{aligned}
\big(g \circ f\big)(x \star_1 y) &= g\big(f(x \star_1 y)\big) && \text{(definition of function composition)} \\
&= g\big(f(x) \star_2 f(y)\big) && (f \text{ is a monoid homomorphism}) \\
&= g\big(f(x)\big) \star_3 g\big(f(y)\big) && (g \text{ is a monoid homomorphism}) \\
&= \big(g \circ f\big)(x) \star_3 \big(g \circ f\big)(y) && \text{(definition of function composition)}
\end{aligned}$$

6.3 Preorders

PreO is a category whose objects are preorders $(P, \leq_P)$ and whose morphisms are monotone functions.

For preorders $(P, \leq_P)$ and $(Q, \leq_Q)$, a function

$$f : P \to Q$$

is *monotone* when, for every $x, y \in P$, $x \leq_P y \implies f(x) \leq_Q f(y)$.

In this category, id and $\circ$ are as in **Set**.

6.4 Partial Orders

Poset is a category whose objects are partial orders and whose morphisms are monotone functions.

Notice that **Poset** is a *subcategory* of **PreO**.

Subcategory

Definition 6. A subcategory $\mathcal{S}$ of a category $\mathcal{C}$ is specified by

- $Obj(\mathcal{S}) \subseteq Obj(\mathcal{C})$ and
- $Arr(\mathcal{S}) \subseteq Arr(\mathcal{C})$

with identities and composition in $\mathcal{S}$ as in $\mathcal{C}$. *Obj* is defined as above, and *Arr* is the set of "arrows" or morphisms in $\mathcal{S}$.

Finite sets

FinSet is the subcategory of **Set** whose objects are *finite* sets.

Bijections

Bij is the subcategory of **FinSet** whose morphisms are all bijections.

Notice that, relative to **Set**, **FinSet** cut down the objects but kept the morphisms; and, relative to **FinSet**, **Bij** cut down morphisms but kept the objects.

Relations

Rel is a category whose objects are sets and whose arrows are relations between sets:

$$\mathbf{Rel}(A, B) = \{ R \mid R \subset A \times B \}$$

Identities are given by the identity relations, relating every element to itself:

$$\mathrm{id}_A = \{ (a, a) \mid a \in A \}$$

and composition is relational composition:

$$R \circ_{A,B,C} S = \left\{ (a, c) \mid \exists b \in B.\, (a, b) \in S \wedge (b, c) \in R \right\} \ .$$

Notice that **Set** is a subcategory of **Rel**. In fact, there is an intermediate category **Pfn** whose morphisms are *partial* functions.

Predicates

Pred is the category whose objects are pairs (S, P) of a set S and a predicate on S, $P \subseteq S$. Its morphisms are functions that respect the predicates, *i.e.*

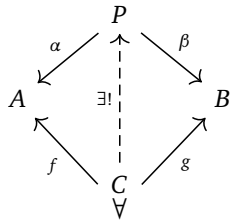
$$(S_1, P_1) \xrightarrow{f} (S_2, P_2)$$

is a function $f : S_1 \to S_2$ such that

$$\forall x \in S.\, x \in P_1 \implies f(x) \in P_2.$$

Identity and composition are as for functions.

Recall the following universal property from the previous discussion:



Before, we stipulated that P, A, B, C were sets and α, β, f, g were functions. However, we can generalize those assertions and instead work in an arbitrary category $\mathcal{C}$; so that P, A, B, C are objects of $\mathcal{C}$ and α, β, f, g are arrows of $\mathcal{C}$ between the respective objects.

In **Set**, we can take $P = A \times B \stackrel{\text{def}}{=} \{(a, b) \mid a \in A \wedge b \in B\}$, $\alpha = \pi_1 : (a, b) \mapsto a$ and $\beta = \pi_2 : (a, b) \mapsto b$ as the final solution. The proof is trivial. Note that in some of the categories that are described in our essay, there will be no solution. In particular, have a look at **Mon**, **PreO**, **Pred**, and **Rel**.