Categories and Quantum Informatics: Categorical semantics

Chris Heunen

Spring 2018

A brief introduction to categorical semantics. We focus in particular on the category **Set** of sets and functions, and the category **Rel** of sets and relations, and present a matrix calculus for relations. We introduce the idea of commuting diagrams, and define isomorphisms, groupoids, skeletal categories, opposite categories and product categories. We then define functors, equivalences and natural transformations, and also products and equalizers.

0.1 Semantics

Suppose `F` and `G` are two fragments of code, and consider the two computer programs:

$$P = (\texttt{if 1 = 1 then F else F})$$
$$Q = (\texttt{if 1 = 1 then F else G})$$

Are P and Q the same programs or not?

Syntactically, they are clearly different: the code fragment P is different than the code fragment Q. But do P and Q compute the same? The answer depends on how fine-grained you are willing to look. From the perspective of machine *operations*, a (nonoptimising) compiler will make P and Q into different executables, because the machine will be storing `G` somewhere in memory when executing Q, even though it is dead code that will never be reached, but not so when executing P. But from perspective of the user, P and Q *behave* the same. They give the same output on every input. They *denote* the same algorithm, namely `F`.

This little analysis we have been doing is called *semantics*. We assigned to the two syntactical fragments P and Q their meaning, in the form of some mathematical objects $[\![P]\!]$ and $[\![Q]\!]$. On the first level, *operational semantics*, that mathematical object encoded all implementation details, so that we could retain the difference between the dead code. On the second level, *denotational semantics*, that mathematical object was a set-theoretical function that transforms input into output, and we didn't care how exactly the function performed that computation.

Denotational semantics is used two show that two programs implement 'the same' algorithm. For example, there are many ways to sort an array, but as long as we don't care about how fast the implementation is they all do the same. Quicksort might be faster than bubble sort, but they both sort. In other words, denotational semantics is used to prove that programs meet their specification, that they 'do what they should do'.

The example with P and Q above was insultingly simple, but you can imagine it gets a lot more complicated when for example recursion is in play. The important thing is that the assignment $P \mapsto [\![P]\!]$ *preserves structure*. For example, if we want to prove something about *concatenating* program fragments, we should have some operation that concatenates their denotations:

$$[\![\texttt{F; G}]\!] = [\![\texttt{G}]\!] \circ [\![\texttt{F}]\!]$$

Similarly, recursion requires us to talk about substitution (calling subprograms with some input) in the syntax, so the semantics had better have some notion of *function space*

$$[\![\texttt{F(X)}]\!] = [\![\texttt{F}]\!]([\![\texttt{X}]\!])$$

where $[\![\texttt{F}]\!]$ can live. For a third example, if the programming language describes concurrent computation, there should be some sort of *parallel composition* of denotations:

$$[\![\texttt{F while G}]\!] = [\![\texttt{F}]\!] \otimes [\![\texttt{G}]\!]$$

The trick is to choose the mathematical object in which the denotations $[\![\texttt{F}]\!]$ live cleverly. It should have the same structure as the programs you're analysing, but abstract away from all the unimportant details, so that powerful mathematical theorems can be used. Popular choices are λ-calculus or partially ordered sets. We will use *categories* as our semantics, which subsumes both.

There is another reason we will use categories. Above we analysed code in an actual programming language. If we consider *quantum* processes, there is no such neat description. The systems are just black boxes that we cannot look inside. But we can still see how the boxes behave; that is precisely the empirical method of physics! Analogously, in computer science, if I give you two executables instead of their source codes, can you still say whether they implement the same algorithm? No, because then you could solve the halting problem.

But you can still analyse the structure of the two computations using denotational semantics. You can see how the *information flows* from input to output and how it is recombined in the process. Categories support this kind of bookkeeping in a beautiful way. As we will see, instead of boring algebra (like in λ-calculus or partially ordered sets), we can manipulate categories graphically completely rigorously, in a way that can even be automated.

In fact, in this way category theory itself becomes a programming language of sorts, where we can click together subroutines in various ways to compose larger programs. In fact, it is unclear what a quantum programming language should look like: if you cannot copy or delete information, you cannot push and pop things on the stack, how do you handle recursion, or even if-then-else statements? Therefore, investigating the categorical semantics first is in a sense the only way to inform the design of a good *quantum programming language.*

0.2 Categories

Categories are formed from two basic structures: *objects* $A, B, C, \ldots$, and *morphisms* $A \xrightarrow{f} B$ going between objects. In this book, we will often think of an object as a *system*, and a morphism $A \xrightarrow{f} B$ as a *process* under which the system A becomes the system B. Categories can be constructed from almost any reasonable notion of system and process. Here are a few examples:

- physical systems, and physical processes governing them;
- data types in computer science, and algorithms manipulating them;
- algebraic or geometric structures in mathematics, and structure-preserving functions;
- logical propositions, and implications between them.

Category theory is quite different from other areas of mathematics. While a category is itself just an algebraic structure — much like a group, ring, or field — we can use categories to organize and understand other mathematical objects. This happens in a surprising way: by neglecting all information about the structure of the objects, and focusing entirely on relationships *between* the objects. Category theory is the study of the patterns formed by these relationships. While at first this may seem limiting, it is in fact empowering, as it becomes a general language for the description of many diverse structures.

Here is the definition of a category.

Definition 0.1. A *category* $\mathbf{C}$ consists of the following data:

- a collection $\mathrm{Ob}(\mathbf{C})$ of *objects*;

- for every pair of objects A and B, a collection $\mathbf{C}(A, B)$ of *morphisms*, with $f \in \mathbf{C}(A, B)$ written $A \xrightarrow{f} B$;
- for every pair of morphisms $A \xrightarrow{f} B$ and $B \xrightarrow{g} C$ with common intermediate object, a *composite* $A \xrightarrow{g \circ f} C$;
- for every object A an *identity morphism* $A \xrightarrow{\mathrm{id}_A} A$.

These must satisfy the following properties, for all objects A, B, C, D, and all morphisms $A \xrightarrow{f} B$, $B \xrightarrow{g} C$, $C \xrightarrow{h} D$:

- *associativity*:

$$h \circ (g \circ f) = (h \circ g) \circ f; \tag{1}$$

- *identity*:

$$\mathrm{id}_B \circ f = f = f \circ \mathrm{id}_A. \tag{2}$$

We will also sometimes use the notation $f \colon A \to B$ for a morphism $f \in \mathbf{C}(A, B)$.

From this definition we see quite clearly that the morphisms are 'more important' than the objects; after all, every object A is canonically represented by its identity morphism id_A. This seems like a simple point, but in fact it is a significant departure from much of classical mathematics, in which particular structures (like groups) play a much more important role than the structure-preserving maps between them (like group homomorphisms.)

Our definition of a category refers to collections of objects and morphisms, rather than sets, because sets are too small in general. The category **Set** defined below illustrates this very well, since Russell's paradox prevents the collection of all sets from being a set. However, such set-theoretical issues will not play a role in this book, and we will use set theory naively throughout. (See the Notes and further reading at the end of this chapter for more sophisticated references on category theory.)

0.3 The category Set

The most basic relationships between sets are given by functions.

Definition 0.2. For sets A and B, a *function* $A \xrightarrow{f} B$ comprises, for each $a \in A$, a choice of element $f(a) \in B$. We write $f \colon a \mapsto f(a)$ to denote this choice.

Writing $\emptyset$ for the empty set, the data for a function $\emptyset \to A$ can be provided trivially; there is nothing for the 'for each' part of the definition to do. So there is exactly one function of this type for every set A. However, functions of type $A \to \emptyset$ cannot be constructed unless $A = \emptyset$. In general there are $|B|^{|A|}$ functions of type $A \to B$, where $|-|$ indicates the cardinality of a set.

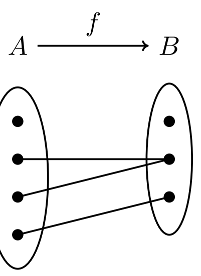
We can now use this to define the category of sets and functions.

Definition 0.3 (**Set**, **FSet**)**.** The category **Set** of sets and functions is defined as follows:

- **objects** are sets $A, B, C, \ldots$;
- **morphisms** are functions $f, g, h, \ldots$;
- **composition** of $A \xrightarrow{f} B$ and $B \xrightarrow{g} C$ is the function $g \circ f \colon a \mapsto g(f(a))$; this is the reason the standard notation $g \circ f$ is not in the other order, even though that would be more natural in some equations such as (5);
- **the identity morphism** on A is the function $\mathrm{id}_A \colon a \mapsto a$.

Define the category **FSet** to be the restriction of **Set** to finite sets.

We can think of a function $A \xrightarrow{f} B$ in a dynamical way, as indicating how elements of A can evolve into elements of B. This suggests the following sort of picture:



(3)

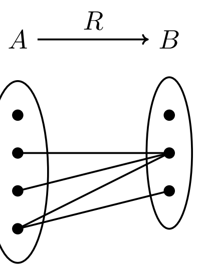
0.4 The category Rel

Relations give a more general notion of process between sets.

Definition 0.4. Given sets A and B, a *relation* $A \xrightarrow{R} B$ is a subset $R \subseteq A \times B$.

If elements $a \in A$ and $b \in B$ are such that $(a, b) \in R$, then we often indicate this by writing $a\,R\,b$, or even $a \sim b$ when R is clear. Since a subset can be defined by giving its elements, we can define our relations by listing the related elements, in the form $a_1\,R\,b_1$, $a_2\,R\,b_2$, $a_3\,R\,b_3$, and so on.

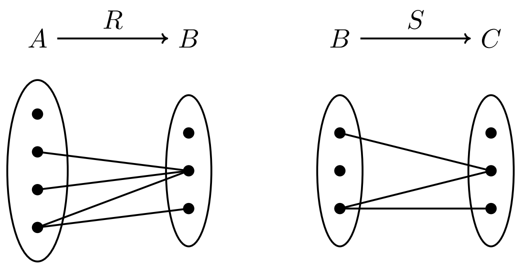
We can think of a relation $A \xrightarrow{R} B$ in a dynamical way, generalizing (3):



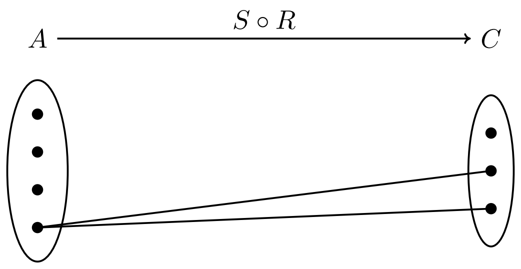
(4)

The difference with functions is that this indicates interpreting a relation as a kind of nondeterministic classical process: each element of A can evolve into any element of B to which it is related. Nondeterminism enters here because an element of A can be related to more than one element of B, so under this interpretation, we cannot predict perfectly how it will evolve. An element of A could also be related to no elements of B: we interpret this to mean that, for these elements of A, the dynamical process halts. Because of this interpretation, the category of relations is important in the study of nondeterministic classical computing.

Suppose we have a pair of relations, with the codomain of the first equal to the domain of the second:



Our interpretation of relations as dynamical processes then suggests a natural notion of composition: an element $a \in A$ is related to $c \in C$ if there is some $b \in B$ with $a\,R\,b$ and $b\,S\,c$. For the example above, this gives rise to the following composite relation:



This definition of relational composition has the following algebraic form:

$$S \circ R := \{(a, c) \mid \exists b \in B \colon aRb \text{ and } bSc\} \subseteq A \times C \tag{5}$$

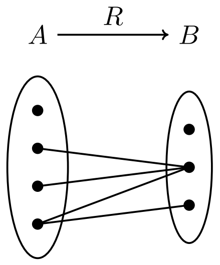
We can write this differently as

$$a\,(S \circ R)\,c \;\Leftrightarrow\; \bigvee_b (bSc \wedge aRb), \tag{6}$$

where $\vee$ represents *logical disjunction* (OR), and $\wedge$ represents *logical conjunction* (AND). Comparing this with the definition of matrix multiplication, we see a strong similarity:

$$(g \circ f)_{ij} = \sum_k g_{ik} f_{kj} \tag{7}$$

This suggests another way to interpret a relation: as a matrix of truth values. For the example relation (4), this gives the following matrix, where we write 0 for false and 1 for true:



$$\rightsquigarrow \qquad \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{8}$$

Composition of relations is then just given by ordinary matrix multiplication, with logical disjunction and conjunction replacing $+$ and $\times$, respectively (so that $1 + 1 = 1$).

There is an interesting analogy between quantum dynamics and the theory of relations. Firstly, a relation $A \xrightarrow{R} B$ tells us, for each $a \in A$ and $b \in B$, whether it is *possible* for a to produce b, whereas a complex-valued matrix $H \xrightarrow{f} K$ gives us the *amplitude* for a to evolve to b. Secondly, relational composition tells us the *possibility* of evolving via an intermediate point, whereas matrix composition tells us the *amplitude* for this to happen.

The intuition we have developed leads to the following definition of the category **Rel**.

Definition 0.5 (**Rel**, **FRel**). The category **Rel** of sets and relations is defined as follows:

- **objects** are sets $A, B, C, \ldots$;
- **morphisms** are relations $R \subseteq A \times B$;

- **composition** of $A \xrightarrow{R} B$ and $B \xrightarrow{S} C$ is the relation

$$\{(a,c) \in A \times C \mid \exists b \in B \colon (a,b) \in R, (b,c) \in S\};$$

- **the identity morphism** on A is the relation $\{(a,a) \in A \times A \mid a \in A\}$.

Define the category **FRel** to be the restriction of **Rel** to finite sets.

While **Set** is a setting for classical physics, and **Hilb** (to be introduced later) is a setting for quantum physics, **Rel** is somewhere in the middle. It seems like it should be a lot like **Set**, but in fact, its properties are much more like those of **Hilb**. This makes it an excellent test-bed for investigating different aspects of quantum mechanics from a categorical perspective.

0.5 Morphisms

It often helps to draw diagrams of morphisms, indicating a way they can be composed. Here is an example:

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "h"'] & B \arrow[r, "g"] \arrow[d, "i"'] & C \\ D \arrow[r, "k"'] & E \arrow[ur, "j"'] & \end{tikzcd} \tag{9}$$

We say a diagram *commutes* when every possible path from one object in it to another is the same. In the above example, this means $i \circ f = k \circ h$ and $g = j \circ i$. It then follows that $g \circ f = j \circ k \circ h$, where we do not need to write parentheses thanks to associativity. Thus we have two ways to speak about equality of composite morphisms: by algebraic equations, or by commuting diagrams.

The following terms are very useful when discussing morphisms.

Definition 0.6 (Domain, codomain, endomorphism, operator)**.** For a morphism $A \xrightarrow{f} B$, its *domain* is the object A, and its *codomain* is the object B. If $A = B$ then we call f an *endomorphism* or *operator*.

Definition 0.7 (Isomorphism, isomorphic)**.** A morphism $A \xrightarrow{f} B$ is an *isomorphism* when it has an *inverse* morphism $B \xrightarrow{f^{-1}} A$ satisfying:

$$f^{-1} \circ f = \mathrm{id}_A \qquad\qquad f \circ f^{-1} = \mathrm{id}_B \tag{10}$$

We then say that A and B are *isomorphic*, and write $A \simeq B$. If only the left equation of (10) holds, f is called *left-invertible.*

Lemma 0.8. *If a morphism has an inverse, it is unique.*

Proof. If g and g' are inverses for f, then

$$g \overset{(2)}{=} g \circ \mathrm{id} \overset{(10)}{=} g \circ (f \circ g') \overset{(1)}{=} (g \circ f) \circ g' \overset{(10)}{=} \mathrm{id} \circ g' \overset{(2)}{=} g'. \qquad \square$$

Example 0.9. Let us see what isomorphisms are like in our example categories:

- in **Set**, the isomorphisms are exactly the bijections of sets;
- in **Rel**, the isomorphisms are the graphs of bijections: a relation $A \xrightarrow{R} B$ is an isomorphism when there is some bijection $A \xrightarrow{f} B$ such that $aRb \Leftrightarrow f(a) = b$;

The notion of isomorphism leads to some important types of category.

Definition 0.10. A category is *skeletal* when any two isomorphic objects are equal.

Definition 0.11 (Groupoid, group)**.** A *groupoid* is a category in which every morphism is an isomorphism. A *group* is a groupoid with one object.

Of course, this definition of group agrees with the ordinary one.

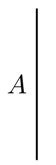
Finally, let us mention some important ways of constructing new categories from given ones.

Definition 0.12. Given a category $\mathbf{C}$, its *opposite* $\mathbf{C}^{\mathrm{op}}$ is a category with the same objects, but with $\mathbf{C}^{\mathrm{op}}(A,B)$ given by $\mathbf{C}(B,A)$. That is, the morphisms $A \longrightarrow B$ in $\mathbf{C}^{\mathrm{op}}$ are morphisms $B \longrightarrow A$ in $\mathbf{C}$.

Definition 0.13. For categories $\mathbf{C}$ and $\mathbf{D}$, their *product* is a category $\mathbf{C} \times \mathbf{D}$, whose objects are pairs (A,B) of objects $A \in \mathrm{Ob}(\mathbf{C})$ and $B \in \mathrm{Ob}(\mathbf{D})$, and whose morphisms are pairs $(A,B) \xrightarrow{(f,g)} (C,D)$ with $A \xrightarrow{f} C$ and $B \xrightarrow{g} D$.

0.6 Graphical notation

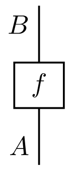
There is a graphical notation for morphisms and their composites. Draw an object A as follows:



(11)

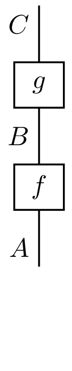
It's just a line. In fact, you should think of it as a picture of the identity morphism $A \xrightarrow{\mathrm{id}_A} A$. Remember: in category theory, the morphisms are more important than the objects.

A morphism $A \xrightarrow{f} B$ is drawn as a box with one 'input' at the bottom, and one 'output' at the top:



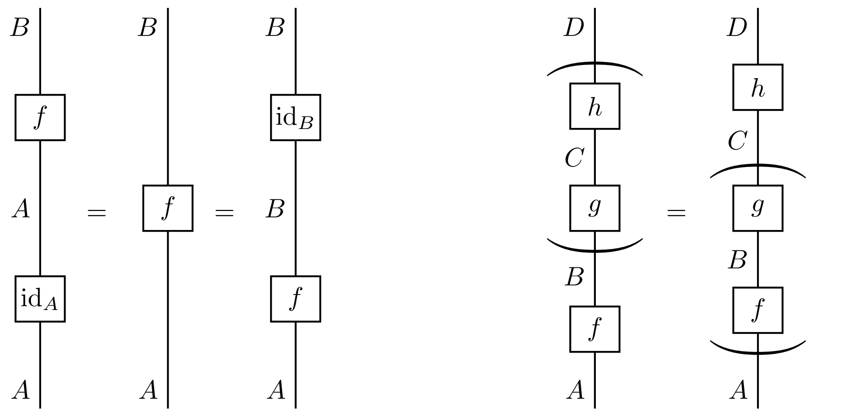
(12)

Composition of $A \xrightarrow{f} B$ and $B \xrightarrow{g} C$ is then drawn by connecting the output of the first box to the input of the second box:



(13)

The identity law $f \circ \mathrm{id}_A = f = \mathrm{id}_B \circ f$ and the associativity law $(h \circ g) \circ f = h \circ (g \circ f)$ then look like:



(14)

To make these laws immediately obvious, we choose to not depict the identity morphisms id_A at all, and not indicate the bracketing of composites.

The graphical calculus is useful because it 'absorbs' the axioms of a category, making them a consequence of the notation. This is because the axioms of a category are about stringing things together in sequence. At a fundamental level, this connects to the geometry of the line, which is also *one-dimensional*. Of course, this graphical representation is quite familiar; we usually draw it horizontally, and call it algebra.

0.7 Functors and natural transformations

Remember the motto that in category theory, morphisms are more important than objects. Category theory takes its own medicine here: there is an interesting notion of 'morphism between categories', as given by the following definition.

Definition 0.14 (Functor, covariance, contravariance)**.** Given categories $\mathbf{C}$ and $\mathbf{D}$, a *functor* $F \colon \mathbf{C} \longrightarrow \mathbf{D}$ is defined by the following data:

- for each object $A \in \mathrm{Ob}(\mathbf{C})$ an object $F(A) \in \mathrm{Ob}(\mathbf{D})$;
- for each morphism $A \xrightarrow{f} B$ a morphism $F(A) \xrightarrow{F(f)} F(B)$ in $\mathbf{D}$.

This data must satisfy the following properties:

- $F(g \circ f) = F(g) \circ F(f)$ for all morphisms $A \xrightarrow{f} B$ and $B \xrightarrow{g} C$ in $\mathbf{C}$;
- $F(\mathrm{id}_A) = \mathrm{id}_{F(A)}$ for every object A in $\mathbf{C}$.

Functors are implicitly *covariant*. There are also *contravariant* versions reversing the direction of morphisms: $F(g \circ f) = F(f) \circ F(g)$. We will only use the above definition, and model the contravariant version $\mathbf{C} \longrightarrow \mathbf{D}$ as (contravariant) functors $\mathbf{C}^{\mathrm{op}} \longrightarrow \mathbf{D}$.

A functor between groups is also called a *group homomorphism*; of course this coincides with the usual notion.

Example 0.15. There is a functor $G \colon \mathbf{Set} \longrightarrow \mathbf{Rel}$ defined by $G(A) = A$ and $G(A \xrightarrow{f} B) = \{(a, f(a) \mid a \in A\}$. It leaves objects alone, and turns functions into (the relation representing) their graph. There is also a functor $P \colon \mathbf{Rel} \longrightarrow \mathbf{Set}$ in the other direction. On objects, $P(A)$ is the powerset of A, and a morphism $A \xrightarrow{R} B$ is turned into the function $P(R) \colon P(A) \longrightarrow P(B)$ given by $X \mapsto \{b \in B \mid \exists a \in X : aRb\}$.

We can use functors to give a notion of equivalence for categories.

Definition 0.16. A functor $F\colon \mathbf{C} \to \mathbf{D}$ is an *equivalence* when it is:

- *full*, meaning that the functions $\mathbf{C}(A, B) \to \mathbf{D}\big(F(A), F(B)\big)$ given by $f \mapsto F(f)$ are surjective for all $A, B \in \mathrm{Ob}(\mathbf{C})$;
- *faithful*, meaning that the functions $\mathbf{C}(A, B) \to \mathbf{D}\big(F(A), F(B)\big)$ given by $f \mapsto F(f)$ are injective for all $A, B \in \mathrm{Ob}(\mathbf{C})$;
- *essentially surjective on objects*, meaning that for each object $B \in \mathrm{Ob}(\mathbf{D})$ there is an object $A \in \mathrm{Ob}(\mathbf{C})$ such that $B \simeq F(A)$.

Just as a functor is a map between categories, so there is a notion of a map between functors, called a *natural transformation.*

Definition 0.17 (Natural transformation, natural isomorphism)**.** Given functors $F\colon \mathbf{C} \to \mathbf{D}$ and $G\colon \mathbf{C} \to \mathbf{D}$, a *natural transformation* $\zeta : F \Longrightarrow G$ is an assignment to every object A in $\mathbf{C}$ of a morphism $F(A) \xrightarrow{\zeta_A} G(A)$ in $\mathbf{D}$, such that the following diagram commutes for every morphism $A \xrightarrow{f} B$ in $\mathbf{C}$.

$$\begin{tikzcd}
F(A) \arrow[r, "\zeta_A"] \arrow[d, "F(f)"'] & G(A) \arrow[d, "G(f)"] \\
F(B) \arrow[r, "\zeta_B"'] & G(B)
\end{tikzcd} \qquad (15)$$

If every component ζ_A is an isomorphism then ζ is called a *natural isomorphism*, and F and G are said to be *naturally isomorphic.*

The notion of natural isomorphism leads to another characterization of equivalence of categories.

Proposition 0.18 (Equivalence by natural isomorphism)**.** *A functor $F\colon \mathbf{C} \to \mathbf{D}$ is an equivalence if and only if there exists a functor $G\colon \mathbf{D} \to \mathbf{C}$ and natural isomorphisms $G \circ F \simeq \mathrm{id}_{\mathbf{C}}$ and $\mathrm{id}_D \simeq F \circ G$.*

Many important concepts in mathematics can be defined in a simple way using functors and natural transformations.

0.8 Products and equalizers

Products and equalizers are recipes for finding objects and morphisms with *universal properties*, with great practical use in category theory. They are special cases of the theory of *limits*, which would form an important part of a typical category theory course.

To get the idea, it is useful to think about the disjoint union $S + T$ of sets S and T. It is not just a bare set; it comes equipped with functions $S \xrightarrow{i_S} S + T$ and $T \xrightarrow{i_T} S + T$ that show how the individual sets embed into the disjoint union. And furthermore, these functions have a special property: a function $S + T \xrightarrow{f} U$ corresponds exactly to a pair of functions of types $S \xrightarrow{f_S} U$ and $T \xrightarrow{f_T} U$, such that $f \circ i_S = f_S$ and $f \circ i_T = f_T$. The concepts of limit and colimit generalize this observation.

Definition 0.19 (Product, coproduct)**.** Given objects A and B, a *product* is an object $A \times B$ together with morphisms $A \times B \xrightarrow{p_A} A$ and $A \times B \xrightarrow{p_B} B$, such that any two morphisms $X \xrightarrow{f} A$ and $X \xrightarrow{g} B$ allow a unique morphism $\binom{f}{g}\colon X \to A \times B$ with $p_A \circ \binom{f}{g} = f$ and $p_B \circ \binom{f}{g} = g$. We can summarize these relationships with the following diagram:

$$\begin{tikzcd} & X \arrow[dl, "f"'] \arrow[d, dashed, "\left(\begin{smallmatrix} f \\ g \end{smallmatrix}\right)"] \arrow[dr, "g"] & \\ A & A \times B \arrow[l, "p_A"] \arrow[r, "p_B"'] & B \end{tikzcd}$$

A *coproduct* is the dual notion, obtained by reversing the directions of all the arrows in this diagram. Given object A and B, a coproduct is an object $A + B$ equipped with morphisms $A \xrightarrow{i_A} A + B$ and $B \xrightarrow{i_B} A + B$, such that for any morphisms $A \xrightarrow{f} X$ and $B \xrightarrow{g} X$, there is a unique morphism $(\, f \; g \,) : A + B \longrightarrow X$ such that $(\, f \; g \,) \circ i_A = f$ and $(\, f \; g \,) \circ i_B = g$.

A category may or may not have products or coproducts (but if they exist they are unique up to isomorphism). In our main example categories they do exist, as we now examine.

Example 0.20. Products and coproducts take the following form in our main example categories:

- in **Set**, products are given by the Cartesian product, and coproducts by the disjoint union;
- in **Rel**, products and coproducts are both given by the disjoint union;

Given a pair of functions $S \xrightarrow{f,g} T$, it is interesting to ask on which elements of S they take the same value. Category theory dictates that we shouldn't ask about elements, but use morphisms to get the same information using a universal property.

Definition 0.21. For morphisms $A \xrightarrow{f,g} B$, their *equalizer* is a morphism $E \xrightarrow{e} A$ satisfying $f \circ e = g \circ e$, such that any morphism $E' \xrightarrow{e'} A$ satisfying $f \circ e' = g \circ e'$ allows a unique $E' \xrightarrow{m} E$ with $e' = e \circ m$:

$$\begin{tikzcd} E \arrow[r, "e"] & A \arrow[r, shift left, "f"] \arrow[r, shift right, "g"'] & B \\ E' \arrow[u, dashed, "m"] \arrow[ur, "e'"'] & & \end{tikzcd}$$

We may think of an equalizer as the largest part of A on which f and g agree. In **Set**, it is given by $\{a \in A \mid f(a) = g(a)\}$. Again, equalizers may or may not exist in a particular category.

Example 0.22. Let's see what equalizers look like in our example categories.

- The category **Set** has equalizers for all pairs of parallel morphisms. An equalizer for $A \xrightarrow{f,g} B$ is the set $E = \{a \in A \mid f(a) = g(a)\}$, equipped with its embedding $E \xrightarrow{e} A$.
- The category **Rel** does not have all equalizers. For example, consider the relation $R = \{(y, z) \in \mathbb{R}^2 \mid y < z \in \mathbb{R}\} \colon \mathbb{R} \longrightarrow \mathbb{R}$. Suppose $E \colon X \longrightarrow \mathbb{R}$ were an equalizer of R and $\mathrm{id}_{\mathbb{R}}$. Then $R \circ R = \mathrm{id}_{\mathbb{R}} \circ R$, so there is a relation $M \colon \mathbb{R} \longrightarrow X$ with $R = E \circ M$. Now $E \circ (M \circ E) = (E \circ M) \circ E = R \circ E = \mathrm{id}_{\mathbb{R}} \circ E = E$, and since $S = \mathrm{id}_X$ is the unique morphism satisfying $E \circ S = E$, we must have $M \circ E = \mathrm{id}_X$. But then xEy and yMx for some $x \in X$ and $y \in \mathbb{R}$. It follows that $y(E \circ M)y$, that is, $y < y$, which is a contradiction.