On the the missing diagrams in Category Theory

Eduardo Ochs

March 11, 2022

Abstract

Most texts on Category Theory are written in a very terse style, in which people pretend a) that all concepts are visualizable, and b) that the readers can reconstruct the diagrams that the authors had in mind based on only the most essential cues. As an outsider I spent years believing that the techniques for drawing diagrams were part of the oral culture of the field, and that the insiders could read texts on CT reconstructing the "missing diagrams" in them line by line and paragraph by paragraph, and drawing for each page of text a page of diagrams with all the diagrams that the authors had omitted. My belief was wrong: there are lots of conventions for drawing diagrams scattered through the literature, but that unified diagrammatic language did not exist. In this chapter I will show an attempt to reconstruct that (imaginary) language for missing diagrams: we will see an extensible diagrammatic language, called DL, that follows the conventions of the diagrams in the literature of CT whenever possible and that seems to be adequate for drawing "missing diagrams" for Category Theory. Our examples include the "missing diagrams" for adjunctions, for the Yoneda Lemma, for Kan extensions, and for geometric morphisms, and how to formalize them in Agda.

Contents

1 Introduction

One of the main themes of this text is a diagrammatic language — let's call it DL — that can be used to draw "missing diagrams" for Category Theory. DL is a *reconstructed language*, and it's easier to explain it if I explain how it was reconstructed, and which of its conventions were improvised. It is easier to do it in the first person.

Suppose that your native language is A and you are learning a language B by a method that includes conversation classes. You will have to improvise a lot, but you will usually get feedback quickly. Now suppose that you are studying a language C — for example, Aeolic Greek ([Car03]) — mostly by yourself, and the corpus of texts in C is small. A good exercise is to try to write your thoughts in C, using loanwords and improvised syntactical constructs when needed, but marking mentally the places in which you had to improvise. In most cases, but not all, you will eventually find ways to rewrite those parts to make them look more like C.

The conventions of DL are explained in sec.2. A few of them don't correspond to anything that I could find in the literature; they are listed at the end of that section.

The best way to introduce DL is to tell the full story of how it evolved from a long sequence of wrong assumptions and from some "thoughts that I wanted to express in DL".

Let me start with some quotes. This one is from Eilenberg and Steenrod ([ES52, p.ix], but I learned it from [Krö07, pp.82–83]):

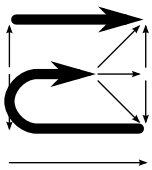
> The diagrams incorporate a large amount of information. Their use provides extensive savings in space and in mental effort. In the case of many theorems, the setting up of the correct diagram is the major part of the proof. We therefore urge that the reader stop at the end of each theorem and attempt to construct for himself the relevant diagram before examining the one which is

> given in the text. Once this is done, the subsequent demonstration can be followed more readily; in fact, the reader can usually supply it himself.

I spent a *lot* of my time studying Category Theory trying to "supply the diagrams myself". In [ES52] supplying the diagrams is not very hard (I guess), but in books like [CWM], in which most important concepts involve several categories, I had to rearrange my diagrams hundreds of times until I reached "good" diagrams...

The problem is that I expected too much from "good" diagrams. The next quotes are from the sections 1 and 12 of an article that I wrote about that ([IDARCT]):

> My memory is limited, and not very dependable: I often have to rededuce results to be sure of them, and I have to make them fit in as little "mental space" as possible...
>
> Different people have different measures for "mental space"; someone with a good algebraic memory may feel that an expression like $\mathsf{Frob} : \Sigma_f(P \wedge f^*Q) \cong \Sigma_f P \wedge Q$ is easy to remember, while I always think diagramatically, and so what I do is that I remember this diagram,

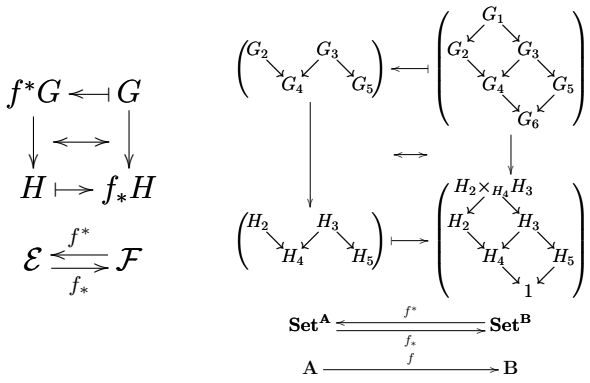


> and I reconstruct the formula from it.

> Let's call the "projected" version of a mathematical object its "skeleton". The underlying idea in this paper is that for the right kinds of projections, and for some kinds of mathetical objects, it should be possible to reconstruct enough of the original object from its skeleton and few extra clues — just like paleontologists can reconstruct from a fossil skeleton the look of an animal when it was alive.

I was searching for a diagrammatic language that would let me express the "skeletons" of categorical definitions and proofs. I wanted these skeletons to be easy to remember — partly because they would have shapes that were easy to remember, and partly because they would be similar to "archetypal cases" ([IDARCT, section 16]).

In 2016 and 2017 I taught a seminar course for undergraduates that covered a bit of Category Theory in the end — see Section 5.5 and [Och19] — and this forced me to invent new techniques for working in two different styles in parallel: a style "for adults", more general, abstract, and formal, and another "for children", with more diagrams and examples. After some semesters, and after writing most of the material that became [PH1], I tried to read again some parts of Johnstone's "Sketches of an Elephant", a book that always felt quite impenetrable to me, and I found a way to present geometric morphisms in toposes to "children". It was based on this diagram,



that we will discuss in detail in 8.12. Its left half is a generic geometric morphism ("for adults"), and its right half is a very specific geometric morphism ("for children") in which everything is easy to understand and to visualize, and that turns out to be "archetypal enough".

I showed that to the few categorists with whom I had contact and the feedback that I got was quite positive. A few of them — the ones who were strictly "adults" — couldn't understand why I was playing with particular cases, and even worse, with finite categories, instead of proving things in the most general case possible, but some others said that these ideas were very nice, that they knew a few bits about geometric morphisms but those bits didn't connect well, and that now they had a family of particular cases to think about, and they had much more intuition than before.

That was the first time that my way of using diagrams yielded something so nice! This was the excuse that I needed to organize a workshop on diagrammatic languages and ways to use particular cases; here's how I advertised it (from [OL18]):

> When we explain a theorem to children — in the strict sense of the term — we focus on concrete examples, and we avoid generalizations, abstract structures and infinite objects.
>
> When we present something to "children", in a wider sense of the term that means "people without mathematical maturity", or even "people without expertise in a certain area", we usually do something similar: we start from a few motivating examples, and then we generalize.
>
> One of the aims of this workshop is to discuss techniques for *particularization* and *generalization*. Particularization is easy; substituing variables in a general statement is often enough to do the job. Generalization is much harder, and one way to visualize how it works is to regard particularization as a projection: a coil projects a circle-like shadow on the ground, and we can ask for ways to "lift" pieces of that circle to the coil continously. *Projections* lose dimensions and may collapse things that were originally different; *liftings* try to reconstruct the missing information in a sensible way. There may be several different liftings for a certain part of the circle, or none. Finding good generalizations is somehow like finding good liftings.
>
> The second of our aims is to discuss *diagrams*. For example, in Category Theory statements, definitions and proofs can be often expressed as diagrams, and if we start with a general diagram and particularize it we get a second diagram with the same shape as the first one, and that second diagram can be used as a version "for children" of the general statement and proof. Diagrams were for a long time considered second-class entities in CT literature ([Krö07] discusses some of the reasons), and were omitted; readers who think very visually would feel that part of the work involved in understanding CT papers and books would be to reconstruct the "missing" diagrams from algebraic statements. Particular cases, even when they were the motivation for the general definition, are also treated as somewhat second-

class — and this inspires a possible meaning for what can call "Category Theory for Children": to start from the diagrams for particular cases, and then "lift" them to the general case. Note that this can be done outside Category Theory too; [Jam01] is a good example.

Our third aim is to discuss *models*. A standard example is that every topological space is a Heyting Algebra, and so a model for Intuitionistic Predicate Logic, and this lets us explain visually some features of IPL. Something similar can be done for some modal and paraconsistent logics; we believe that the figures for that should be considered more important, and be more well-known.

This is from the second announcement:

If we say that categorical definitions are "for adults" - because they may be very abstract - and that particular cases, diagrams, and analogies are "for children", then our intent with this workshop becomes easy to state. "Children" are willing to use "tools for children" to do mathematics, even if they will have to translate everything to a language "for adults" to make their results dependable and publishable, and even if the bridge between their tools "for children" and "for adults" is somewhat defective, i.e., if the translation only works on simple cases...

We are interested in that *bridge* between maths "for adults" and "for children" in several areas. Maths "for children" are hard to publish, even informally as notes (see this thread

http://angg.twu.net/categories-2017may02.html

in the Categories mailing list), so often techniques are rediscovered over and over, but kept restricted to the "oral culture" of the area.

Our main intents with this workshop are:

- to discuss (over coffe breaks!) the techniques of the "bridge" that we currently use in seemingly ad-hoc ways,
- to systematize and "mechanize" these techniques to make them quicker to apply,

- to find ways to publish those techniques — in journals or elsewhere,
- to connect people in several areas working in related ideas, and to create repositories of online resources.

In the UniLog 2018 I was able to chat with several categorists, and they told me that the oral culture of CT was not as I was expected: if there are standard ways to draw diagrams they are not widely known. I also spent two evenings with Peter Arndt working on a certain factorization of geometric morphisms "for children" — and this made me feel that at some point I would be able to present applications of this diagrammatic language in "top-tier" conferences that would not accept works with holes.

The following quote is from the abstract of my submission ([MDE]) to the ACT2019:

> Imagine two category theorists, Aleks and Bob, who both think very visually and who have exactly the same background. One day Aleks discovers a theorem, T_1, and sends an e-mail, E_1, to Bob, stating and proving T_1 in a purely algebraic way; then Bob is able to reconstruct by himself Aleks's diagrams for T_1 exactly as Aleks has thought them. We say that Bob has reconstructed the *missing diagrams* in Aleks's e-mail.
>
> Now suppose that Carol has published a paper, P_2, with a theorem T_2. Aleks and Bob both read her paper independently, and both pretend that she thinks diagrammatically in the same way as them. They both "reconstruct the missing diagrams" in P_2 in the same way, even though Carol has never used those diagrams herself.

and this from my submission ([Och20]) to Diagrams 2020:

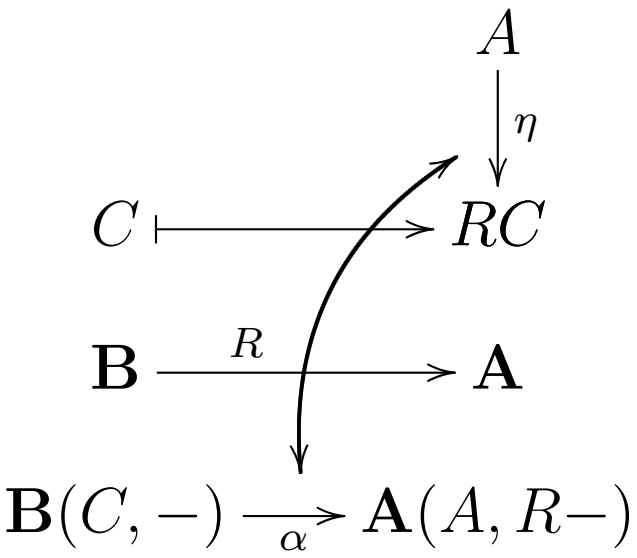
> Category Theory gives the impression of being an area where most concepts and arguments are stated and formalized via diagrams, but this is not exactly true... in most texts almost everything is done algebraically, and the reader is expected to be able to reconstruct the "missing diagrams" by himself.
>
> I used to believe, as an outsider, that some people who grew up immersed the oral culture of the area would know several techniques for "drawing the missing diagrams". My main intent when

I organized the workshop "Logic for Children" at the UniLog 2018 [OL18] was to collect some of these folklore techniques, compare them with the ones that I had developed myself to study CT, and formalize them all — but what I found instead was that everybody that I could get in touch with used their own ad-hoc techniques, and that what I was trying to do was either totally new to them, or at least new in its level of detail.

The story will continue at the end of sec.2, after the list of conventions.

2 The conventions

The conventions that I will present now are the ones that we will need to interpret the diagram below, that is essentially the Proposition 1 in the proof of the Yoneda Lemma in [CWM, Section III.2]; we will call that diagram the "Basic Example" and also "diagram Y0". In the sections 8–8.12 we will see how extend DL to make it able to interpret the diagram for geometric morphisms of the Introduction.



(CD) Our diagrams are made of components that are nodes and arrows. The nodes can contain arbitrary expressions. The arrows work as connectives, and each arrow can be interpreted as the top-level connective in

the smallest subexpression that contains it. For example, the curved arrow in the diagram above can be interpreted as:

$$(A \xrightarrow{\eta} RC) \leftrightarrow (\mathbf{B}(C,-) \xrightarrow{T} \mathbf{A}(A, R-)).$$

(C→) Arrows that look like '→' ("`\to`") represent hom-sets, or, in **Set**, spaces of functions. When a '→' arrow is named the name stands for an element of that hom-set. For example, in $A \xrightarrow{\eta} RC$ we have $\eta : A \to RC$.

(C↦) Arrows that look like '↦' ("`\mapsto`") represent internal views of functions or functors. This has some subtleties; see Section 5.

(C↔) Arrows that look like '↔' ("`\leftrightarrow`") represent bijections or isomorphisms.

(CAI) "Above" usually means "inside", or "internal view". In the diagram above the morphism $\eta : A \to RC$ is in $\mathbf{A}$ and C is an object of $\mathbf{B}$. Also, the arrow $C \mapsto RC$ is above $\mathbf{B} \xrightarrow{R} \mathbf{A}$, and this means that it is an internal view of the functor R. Note that *usually* is not *always* — and $\mathbf{B} \xrightarrow{R} \mathbf{A}$ is not an internal view of $\mathbf{B}(C,-) \xrightarrow{T} \mathbf{A}(A, R-)$.

(CO) When the definition of a component of our diagram is "obvious" in the sense of "there is a unique natural construction for an object with that name", we will usually omit its definition and *pretend* that it is obvious; same for its uniqueness. See Section 3.

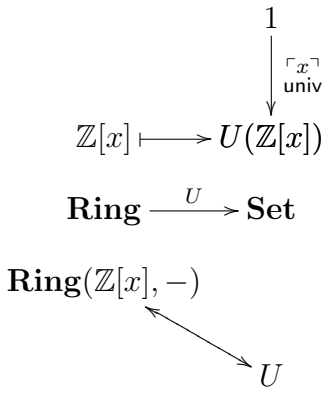
(CC) Everything commutes by default, and non-commutative cells have to be indicated explicitly. See Section 4.

(CTL) The default "meaning" for a diagram *without quantifiers* is the definition of its top-level component. There is a natural partial order on the components of a diagram, in which $\alpha \prec \beta$ iff α is "more basic" than β, or, in other words, if α needs to be defined before β. In the diagram above the top-level component is the curved bijection.

(CMQ) The default "meaning" for a diagram with quantifiers is a proposition. See Sections 4–4.2 for how to obtain that proposition.

(CAdj) *I use shapes based on my way of drawing adjunctions whenever possible.* I like adjunctions so much that when I want to explain Category Theory to someone who knows just a little bit of Maths I always start by the adjunction $(\times B) \dashv (B\to)$ of Section 5.4; I always draw it in a canonical way, with the left adjoint going left, the right adjoint going right, and the morphisms going down. In Proposition 1 of [CWM, Section III.2] the map η is a universal arrow, and someone who learns adjunctions first sees the unit maps $\eta : A \to (B\to(A\times B))$ as the first examples of universal arrows — so that's why the upper part of the diagram above is drawn in this position.

(CYo) *I use shapes based on my way of drawing the Yoneda Lemma whenever possible.* Look at the sections 7–7.3 and 8.4–8.6: the upper parts of their diagrams look like parts of adjunctions, but the other parts do not. For example, I draw "The functor U : **Ring** $\to$ **Set** is representable" as:

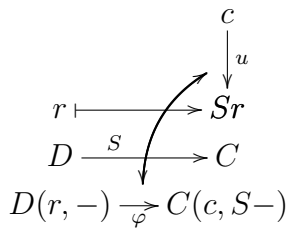


(CDT) A diagram acts a dictionary of default types for symbols. See sec.6.5.

(CIA) Default types allow us to use indefinite articles in a precise way. An example: we have $\eta : \mathrm{Hom}_{\mathbf{A}}(A, RC)$, so "an η" means "an element of $\mathrm{Hom}_{\mathbf{A}}(A, RC)$". See sec.6.6.

(COT) We use a notation as close to the original text as possible, especially when we are trying to draw the missing diagrams for some existing text. If we were drawing the missing diagrams for the Proposition 1 of

[CWM, Section III.2] our diagram would be this:



but I hate Mac Lane's choice of letters, so I decided to use another notation here.

(CSk) Suppose that we have a piece of text — say, a paragraph P — and we want to reconstruct the "missing diagram" D for P. Ideally this D should be a "skeleton" for P, in the sense that it should be possible to reconstruct the ideas in P from the diagram D using very few extra hints; see [IDARCT, sec.12].

(CTT) Our diagrams should be close to Type Theory: it should be possible to use them as a scaffolding for formalizing our text in (pseudocode for) a proof assistant.

(CFSh) The image by a functor of a diagram D is drawn with the same shape as D.

(CISh) The internal view of a diagram D is drawn with the same shape as D, modulo duplications — see section 5.

(CPSh) A particular case of a diagram D is drawn with the same shape as D.

(CNSh) A translation of a diagram D to another notation is drawn with the same shape as D.

The conventions (CD)–(CMQ) and (CFSh)–(CNSh) all appear in diagrams in [MacLaneNotes], [Freyd76], [FS90], [Tay99], [Riehl], [Leinster], but very few of them are spelled out explicitly, and the idea of "same shape" is never stressed. See [NG14, p.179] for a neat example of "substitution produces something with the same shape" and [Penrose] for a language for drawing diagrams from high-level specifications in which it may be possible to implement the rules about "same shape".

The other conventions *may* be new, but remember from the introduction that most of the work on diagrammatic languages lies below the threshold of publishability... so conventions corresponding to those may be folklore knowledge in groups that I don't have contact with *yet*.

The convention (COT) is obvious in retrospect, but giving a name to it saved me from my tendency to invent new notations. The conventions (CDT) and (CIA) replace the idea of downcasings from [IDARCT, sec.3], that didn't work well. Sections 8–8.12 show how to add new conventions to DL, and sec.8.3 shows that we can add a bad convention and mark it as a temporary hack.

There are many notations for Type Theory. To make this chapter more readable in the convention (CTT) I will use a pseudocode that is halfway between standard mathematical notation and Agda; the companion paper [Och22] will show how to translate it to real Agda (with the library [HC20]).

Most texts on CT use diagrams to *prove* theorems. Here will use them to *understand* theorems, and to translate between languages. Our approach can be seen as an extension of [Gan13] to Category Theory; see also [JIB22], that is a recent book that follows many of the ideas in [Gan13].

3 Finding "the" object with a given name

One of the books that I tried to read when I was starting to learn Category Theory was Mac Lane's [CWM]. It is written for readers who know a lot of mathematics and who can follow some steps that it treats as obvious. I was not (yet) a reader like that, but I wanted to become one.

There is one specific thing that [CWM] does pretending that it is obvious that I found especially fascinating. It "defines" functors by describing their actions on objects, and it leaves to the reader the task of discovering their actions on morphisms. Let's see how to find these actions on morphisms.

A functor $F : \mathbf{A} \to \mathbf{B}$ has four components:

$$F = (F_0, F_1, \mathsf{respids}_F, \mathsf{respcomp}_F).$$

They are its action on objects, its action on morphisms, the assurance that it takes identity maps to identity maps, and the assurance that it respects compositions. When Mac Lane says this,

> Fix a set B. Let $(\times B)$ denote *the* functor that takes each set A to $A \times B$.

he is saying that $(\times B)_0 A = A \times B$, or, more precisely, this:

$$(\times B)_0 := \lambda A.\, A \times B$$

The "*the*" in the expression "Let $(\times B)$ denote *the* functor..." implies that the precise meaning of $(\times B)_1$ is easy to find, and that it is easy to prove $\mathsf{respids}_{(\times B)}$ and $\mathsf{respcomp}_{(\times B)}$.

If $f : A' \to A$ then $(\times B)_1 f : (\times B)_0 A' \to (\times B)_0 A$. We know the *name* of the image morphism, $(\times B)_1 f$, and its *type*,

$$(\times B)_1 f : A' \times B \to A \times B,$$

and it is implicit that there is an "obvious" natural construction for this $(\times B)_1 f$ from f. A natural construction is — TA-DAAAA!!! — a λ-term, so we are looking for a term of type $A' \times B \to A \times B$ that can be constructed from $f : A' \to A$.

In a big diagram:

$$\frac{f : A' \to A}{(\times B)_1 f : A' \times B \to A \times B} \quad \Rightarrow \quad \dfrac{\dfrac{\dfrac{[p : A' \times B]^1}{\pi p : A' \qquad f : A' \to A}}{f(\pi p) : A} \qquad \dfrac{[p : A' \times B]^1}{\pi' p : B}}{(f(\pi p), \pi' p) : A \times B}\ 1}{(\lambda p{:}A' \times B.(f(\pi p), \pi' p) : A' \times B \to A \times B}$$

A double bar in a derivation means "there are several omitted steps here", and sometimes a double bar suggests that these omitted steps are obvious. The derivation on the left says that there is an "obvious" way to build a $(\times B)_1 f : A' \times B \to A \times B$ from a "hypothesis" $f : A' \to A$. If we expand its double bar we get the tree at the right, that shows that the "precise meaning" for $(\times B)_1 f$ is $(\lambda p{:}A' \times B.(f(\pi p), \pi' p)$. More formally (and erasing a typing),

$$(\times B)_1 := \lambda f.(\lambda p.(f(\pi p), \pi' p)).$$

The expansion of the double bar above becomes something more familiar if we translate the trees to Logic using Curry-Howard:

$$\frac{P \to Q}{P \wedge R \to Q \wedge R} \quad \Rightarrow \quad \dfrac{\dfrac{\dfrac{[P \wedge R]^1}{P \qquad P \to Q}}{Q} \qquad \dfrac{[P \wedge R]^1}{R}}{Q \wedge R}\ 1}{P \wedge R \to Q \wedge R}$$

We obtain the tree at the right by *proof search*.

Let's give a name for the operation above that obtained a term of type $A' \times B \to A \times B$: we will call that operation *term search*, or, as it is somewhat related to type inference, *term inference*.

Term search may yield several different construction and trees, and so several non-equivalent terms of the desired type. When Mac Lane says "*the* functor $(\times B)$" he is indicating that:

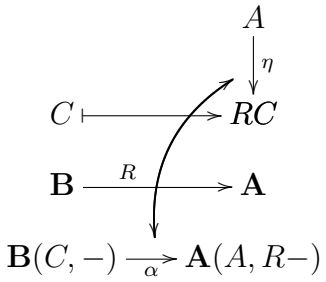
- a term for $(\times B)_1$ is easy to find (note that we use the expression "a *precise meaning* for $(\times B)_1$"),
- all other natural constructions for something that "deserves the name" $(\times B)_1$ yield terms equivalent to that first, most obvious one,
- proving $\mathsf{respids}_{(\times B)}$ and $\mathsf{respcomp}_{(\times B)}$ is trivial.

In many situations we will start by just the name of a functor, as the "$(\times B)$" in the example above, and from that name it will be easy to find *the* "precise meaning" for $(\times B)_0$, and from that the "precise meaning" for $(\times B)_1$, and after that proofs that $\mathsf{respids}_{(\times B)}$ and $\mathsf{respcomp}_{(\times B)}$. We will use the expression "...deserving the name..." in this process — terms for $(\times B)_0$, $(\times B)_1$, $\mathsf{respids}_{(\times B)}$, and $\mathsf{respcomp}_{(\times B)}$ "deserve their names" if they obey the expected constraints.

For a more thorough discussion see [IDARCT].

These ideas of "finding a precise meaning" and "finding (something) deserving that name" can also be applied to morphisms, natural transformations, isomorphisms, and so on.

In Section 7.2 we will see how to find natural constructions for the two directions of the bijection in the Basic Example — or how the expand the double bars in the two derivations here:

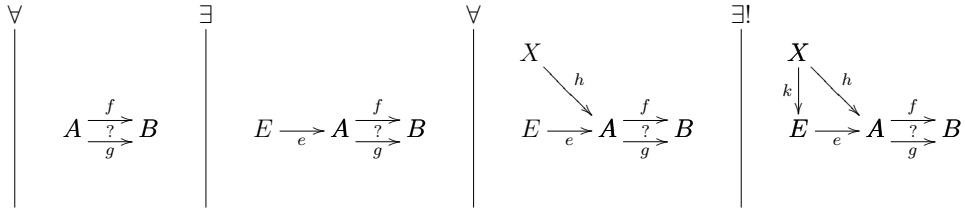


$$\frac{\eta : A \to RC}{\alpha : \mathbf{B}(C,-) \to \mathbf{A}(A,R-)} \qquad \frac{\alpha : \mathbf{B}(C,-) \to \mathbf{A}(A,R-)}{\eta : A \to RC}$$

I am not aware of any papers or books on CT that discuss how to (re)construct a functor from its action on objects or from its name, but Agda has a tool that can be used for that: look for the section on "Automatic Proof Search" in [AgdaMan].

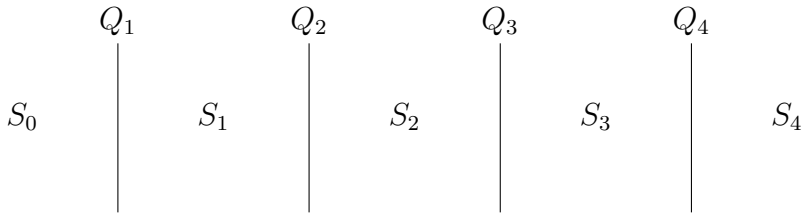
4 Freyd's diagrammatic language

In [Freyd76] Peter Freyd presents a very nice diagrammatic language that can be used to express *some* definitions from Category Theory. For example, this is the statement that a category has all equalizers:



All cells in these diagrams commute by default, and non-commuting cells have to be indicated with a '?'. Each vertical bar with a '∀' above it means "for all extensions of the previous diagram to this one such that everything commutes"; a vertical bar with a '∃!' above it means "there exists a unique extension of the previous diagram to this one such that everything commutes", and so on. See the scan in [Freyd76] for the basic details of how to formalize these diagrams, and the book [FS90, p.28 onwards], for tons of extra details, examples, and applications.

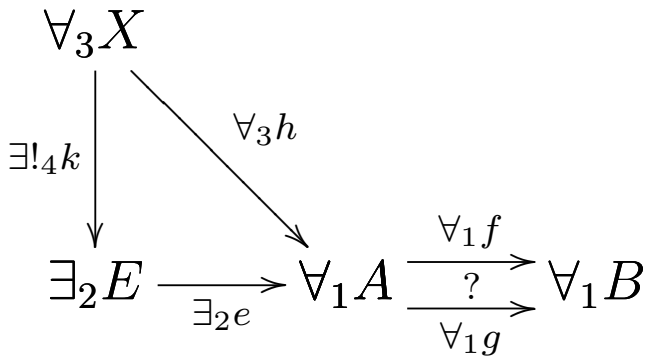
Let's call the subdiagrams of a diagram like the one above its "stages". Its stage 0 is empty, its stage 1 has two objects and two arrows, its last stage has four objects and five arrows, and the quantifiers separating the stages are $Q_1 = \forall$, $Q_2 = \exists$, $Q_3 = \forall$, $Q_4 = \exists!$. They are structured like this:



I was not very good at drawing all stages separately — it was boring, it took me too long, and I often got distracted and committed errors — so I started to play with extensions of that diagrammatic language.

4.1 Adding quantifiers

Here is a simple way to draw all stages at once. We start from a diagram for the "last stage with quantifiers", that we will call LSQ:



We can recover all the stages and quantifiers from it. The numbered quantifiers in it are $\forall_1$, $\exists_2$, $\forall_3$, and $\exists!_4$. The highest number in them is 4, so we set $n = 4$ (n is the index of the last stage), and we set "stage 4 with quantifiers", SQ_4, to LSQ. To obtain the SQ_3 from SQ_4 we delete all nodes an arrows in SQ_4 that are annotated with a '$\exists!_4$'; to obtain SQ_2 from SQ_3 we delete all nodes an arrows in SQ_3 that are annotated with a '$\forall_3$', and so on until we get a diagram SQ_0, that in this example is empty. To obtain each S_k — a stage in the original diagrammatic language from Freyd, that doesn't have quantifiers — from the corresponding SQ_k we treat all the quantifiers in SQ_k as mere annotations, and we erase them; for example, '$\exists_2 e$' becomes 'e', and $\forall_1 A$ becomes A. To obtain the quantifiers Q_1, Q_2, Q_3, Q_4 that are put in the vartical bars that separate the stages, we just assign $\forall_1$, $\exists_2$, $\forall_3$, and $\exists!_4$ to them, without the numbers in the subscripts.

Bonus convention: when the quantifiers in a diagram are just '$\forall$'s and '$\exists!$'s without subscripts the '$\forall$'s are to be interpreted as '$\forall_1$' and the '$\exists!$'s as '$\exists!_2$'s.

4.2 Adding functors

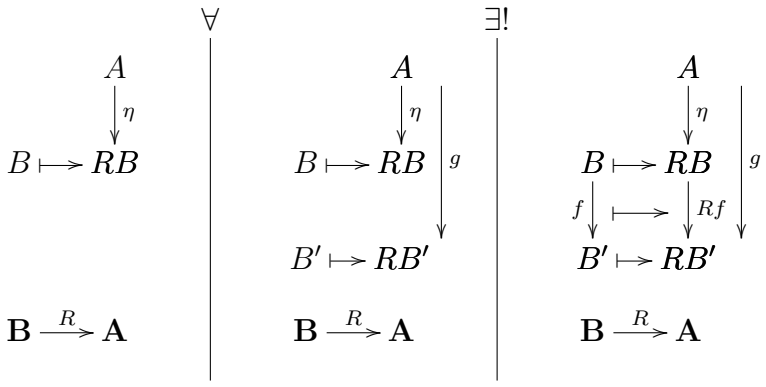
Freyd's language can't represent functors — in the sense of diagrams like the ones in sec.5.2 — and I wanted to use it to draw the missing diagrams for definitions involving functors, so I had to extend it again.

Let me use an example to discuss this. This is the definition of universal arrow in [CWM, p.55], including the original diagram, modulo change of letters:

> **Definition.** If $R : \mathbf{B} \to \mathbf{A}$ is a functor and A an object of $\mathbf{A}$, a universal arrow from A to R is a pair (B, η) consisting of an object B of $\mathbf{B}$ and and arrow $\eta : A \to RB$ of $\mathbf{A}$ such that to every pair (B', g) with B' an object of $\mathbf{B}$ and $g : A \to RB'$ an arrow of $\mathbf{A}$, there is a unique arrow $f : B \to B'$ of $\mathbf{B}$ with $Rf \circ \eta = g$. In other words, every arrow h to R factors uniquely through the universal arrow η, as in the commutative diagram:

$$\begin{array}{ccc} A & \xrightarrow{\eta} & RB \\ \| & & \downarrow{\scriptstyle Rf} \\ A & \xrightarrow{g} & RB', \end{array} \qquad \begin{array}{c} B \\ \downarrow{\scriptstyle f} \\ B'. \end{array}$$

The definition itself goes only up to the "with $Rf \circ \eta = g$.", so let me ignore the part starting from "In other words", and draw a better "missing diagram" for the definition:



This diagram is quite close to being a skeleton for the definition of universal arrow. It can be interpreted as a proposition, and the only extra hint

that we need is that "universalness" for the arrow η corresponds to the truth of that proposition. Here's how to extract the proposition from it:

In a context where:	$\mathbf{A}$ is a category,
	$\mathbf{B}$ is a category,
	$R : \mathbf{B} \to \mathbf{A}$,
	$A \in \mathbf{A}$,
	$B \in \mathbf{B}$,
	$\eta : A \to RB$,
for all	$B' \in \mathbf{B}$ and
	$g : A \to RB'$,
there exists a unique	$f : B \to B'$ such that
	$Rf \circ \eta = g$.

To convert that to a definition of universalness we just have to replace the "for all" by "(B, η) is a universal arrow for A to R iff for all".

The convention for quantifiers from sec.4.1 lets us rewrite the diagram in three stages above as:

$$\begin{array}{ccc} A & & \\ \downarrow{\scriptstyle \eta} & & \\ B \longmapsto RB & & \forall g \\ \exists! f \downarrow \quad \longmapsto \quad \downarrow Rf & & \downarrow \\ \forall B' \mapsto RB' & & \end{array}$$

$$\mathbf{B} \xrightarrow{R} \mathbf{A}$$

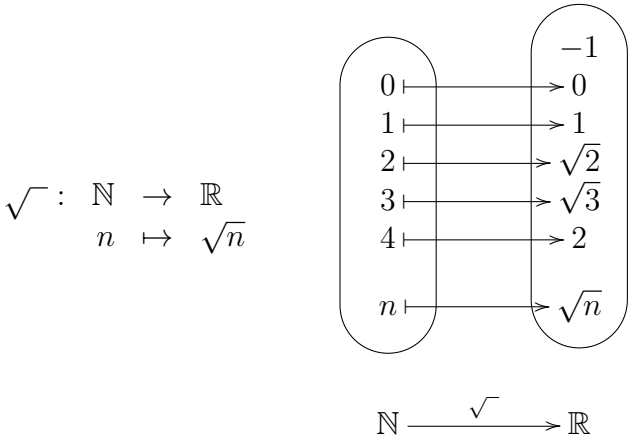
Also, I noticed that I could omit most typings when they could be inferred from the diagram. I could "formalize" the diagram above as: "in a context where $(\mathbf{A}, \mathbf{B}, R, A, B, \eta)$ are as in the diagram above, we say that (B, η) is a universal arrow from A to R when $\forall(B', g).\exists! f.(Rf \circ \eta = g)$". This looked too loaded to be used in public, but it was practical for private notes — and I could even omit the "$Rf \circ \eta = g$", as everything commutes by default. In sec.6.5 we will see a way to formalize this method for omitting

and reconstructing types, and in sec.6.6 we will see a second way to define universalness.

Note that when we erase a node or arrow we also erase everything that depends on it. In the example above SQ_2 has an arrow labeled $\exists!_2 f$; to obtain SQ_1 from SQ_2 we have to erase that arrow, the arrow Rf, and the arrow $f \mapsto Rf$ — and to obtain SQ_0 from SQ_1 we have to erase the arrow g, the node B', the node RB', and the arrow $B' \mapsto RB'$.

5 Internal views

My favorite way of introducing internal views is with the diagram below:



The parts with the two blobs and '$\longmapsto$'s between them is based on how I was taught sets and functions when I was a kid; it is an internal view of the $\mathbb{N} \xrightarrow{\sqrt{\ }} \mathbb{R}$ below it. Not all elements of $\mathbb{N}$ are shown in the blob-view of $\mathbb{N}$, but the ones that are shown are named; compare this with [LR03, p.2 onwards], in which the elements are usually dots.

The arrow $n \longmapsto \sqrt{n}$ between the blobs shows a *generic element* of $\mathbb{N}$ and its image, and the other '$\longmapsto$'s are *substitution instances of it*, like this:

$$(n \longmapsto \sqrt{n})[n := 2] = (2 \longmapsto \sqrt{2})$$

In some cases, like $4 \longmapsto 2$, we write 2 instead of $\sqrt{4}$ because $\sqrt{4}$ "reduces to" 2, as explained in the next section.

5.1 Reductions

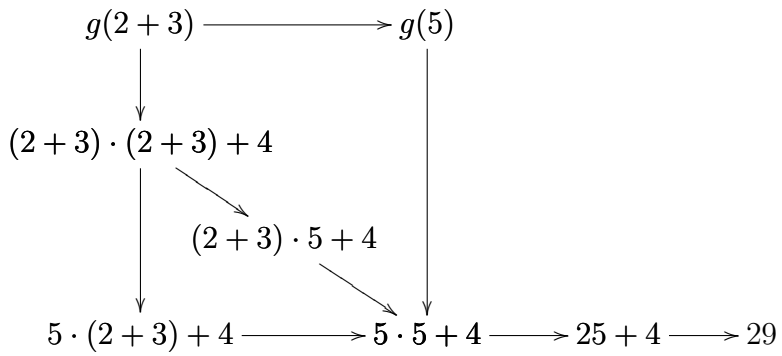
The convention (C↦) says that an arrow $\alpha \mapsto \beta$ above an arrow $A \xrightarrow{f} B$ should be interpreted as meaning $f(\alpha) \rightsquigarrow \beta$, where '$\rightsquigarrow$' means "reduces to"; the standard example is $\sqrt{4} \rightsquigarrow 2$. In a diagram:

$$\begin{array}{ll} 4 \longmapsto 2 & \sqrt{4} \rightsquigarrow 2 \\ n \longmapsto \sqrt{n} & \end{array} \qquad \begin{array}{ll} \alpha \longmapsto \beta & f(\alpha) \rightsquigarrow \beta \\ A \xrightarrow{f} B & \end{array}$$

$$\mathbb{N} \xrightarrow{\sqrt{\ }} \mathbb{R}$$

The idea of reduction comes from λ-calculus. We write $\alpha \overset{1}{\rightsquigarrow} \beta$ to say that the term α reduces to β in one step, and $\alpha \overset{*}{\rightsquigarrow} \gamma$ to say that there is a finite sequence of one-step reductions that reduce α to γ. Here we are interested in reduction in a system with constants, in which for example $(\sqrt{\ })(4) \overset{1}{\rightsquigarrow} 2$.

Here is a directed graph that shows all the one-step reductions starting from $g(2+3)$, considering $g(a) = a \cdot a + 4$:

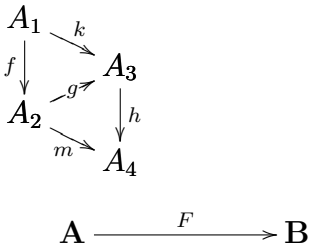


Note that all reductions sequences starting from $g(2+3)$ terminate at the same term, 29 — "the term $g(2+3)$ is strongly normalizing" —, and reduction sequences from $g(2+3)$ may "diverge" but they "converge" later — this is the "Church-Rosser Property", a.k.a. "confluence".

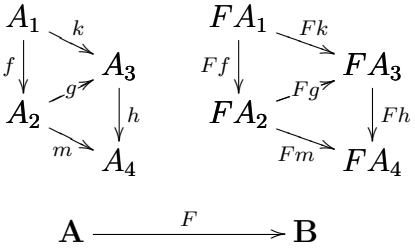
A good place to learn about reduction in systems with constants is [SICP].

5.2 Functors

By the convention (CFSh) the image of the diagram above **A** in the diagram below — remember that *above* usually means *inside* —

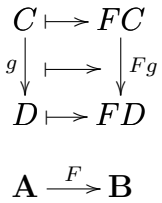


is a diagram with the same shape over **B**. We draw it like this:



In this case we don't draw the arrows like $A_1 \mapsto FA_1$ because there would be too many of them — we leave them implicit.

We say that the diagram above is *an* internal view of the functor F. To draw *the* internal view of the functor $F : \mathbf{A} \to \mathbf{B}$ we start with a diagram in **A** that is made of two generic objects and a generic morphism between them. We get this:



Compare this with the diagram with blob-sets in Section 5, in which the '$n \mapsto \sqrt{n}$' says where a generic element is taken.

Any arrow of the form $\alpha \mapsto \beta$ above a functor arrow $\mathbf{A} \xrightarrow{F} \mathbf{B}$ is interpreted as saying that F takes α to β, or, in the terminology of the section 5.1, that

$F\alpha$ reduces to β. So this diagram

$$\begin{array}{ccc} B & \longmapsto & A{\times}B \\ {\scriptstyle f}\downarrow & \longmapsto & \downarrow{\scriptstyle \lambda p.(\pi p, f(\pi' p))} \\ C & \longmapsto & A{\times}C \end{array}$$

$$\mathbf{Set} \xrightarrow{(A\times)} \mathbf{Set}$$

defines $(A\times)$ as:

$$\begin{array}{lcl} (A\times)_0 & := & \lambda B.\, A \times B, \\ (A\times)_1 & := & \lambda f.\lambda p.(\pi p, f(\pi' p)). \end{array}$$

In this case we can also use internal views of $(A\times)$ to define $(A\times)_1$:

$$\begin{array}{ccccc} B & \longmapsto & A{\times}B & (a,b) & p \\ {\scriptstyle f}\downarrow & \longmapsto & \downarrow{\scriptstyle (A\times)f} & \top\downarrow & \top\downarrow \\ C & \longmapsto & A{\times}C & (a, f(b)) & (\pi p, f(\pi' p)) \end{array}$$

$$\mathbf{Set} \xrightarrow{(A\times)} \mathbf{Set}$$

5.3 Natural transformations

Suppose that we have two functors $F, G : \mathbf{A} \to \mathbf{B}$ and a natural transformation $T : F \to G$. A first way to draw an internal view of T is this:

$$\begin{array}{ccc} & & FC \\ C & \begin{array}{c}\nearrow \\ \to \\ \searrow\end{array} & \downarrow{\scriptstyle TC} \\ & & GC \end{array}$$

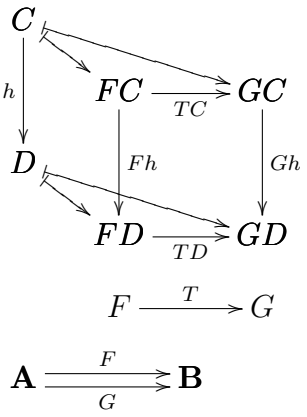
$$\mathbf{A} \underset{G}{\overset{F}{\rightrightarrows}} \mathbf{B}$$

If we start with a morphism $h : C \to D$ in $\mathbf{A}$, like this,

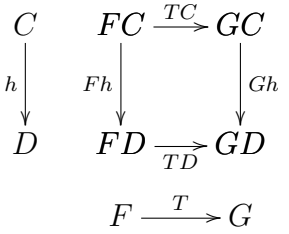
$$\begin{array}{c} C \\ {\scriptstyle h}\downarrow \\ D \end{array}$$

$$\mathbf{A} \underset{G}{\overset{F}{\rightrightarrows}} \mathbf{B}$$

the convention (CFSh) would yield an image of h by F and another by G, and we can draw the arrows TC and TD to obtain a commuting square in **B**:

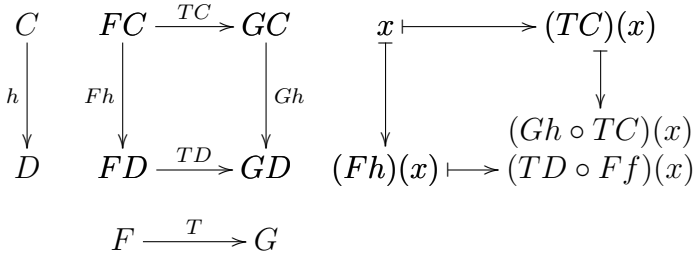


This way of drawing internal views of natural transformations yields diagrams that are too heavy, so we will usually draw them as just this:



Note that the input morphism is at the left, and above $F \xrightarrow{T} G$ we draw its images by F, G, and T.

When the codomain of F and G is **Set** we will sometimes also draw at the right an internal view of the commuting square, like this:

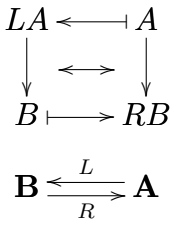


Then the commutativity of the middle square is equivalent to $\forall x \in FC.(Gh \circ TC)(x) = (TD \circ Ff)(x)$. Note that in this case the square at the right is an internal view of an internal view.

In Section 3 we saw that a functor has four components. A natural transformation has two: $T = (T_0, \mathsf{sqcond}_T)$, where T_0 is the operation $C \mapsto TC$ and sqcond_T is the guarantee that all the induced squares commute.

5.4 Adjunctions

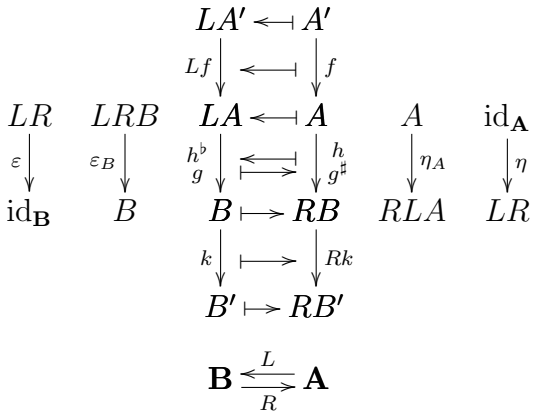
We will draw adjunctions like this,



with the left adjoint going left and the right adjoint going right. My favorite names for the left and right adjoints are L and R. The standard notation for that adjunction is $L \dashv R$.

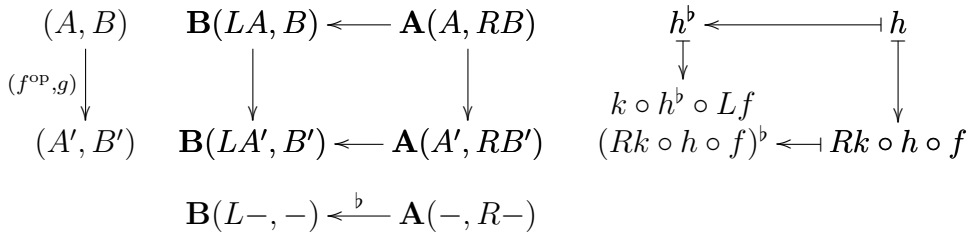
The top-level component of the diagram above is the bijection arrow in the middle of the square — it says that $\mathrm{Hom}(LA, B) \leftrightarrow \mathrm{Hom}(A, RB)$. It is implicit that we have bijections like that for all A and B; it is also implicit that that bijection is natural in some sense.

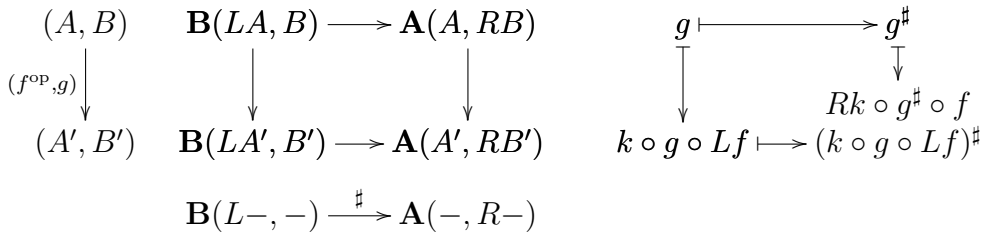
We will sometimes expand adjunction diagrams by adding unit and counit maps, the unit and the unit as natural transformations, the actions of L and R on morphisms, and other things. For example:



We can obtain the naturality conditions by regarding $\flat$ and $\sharp$ as natural

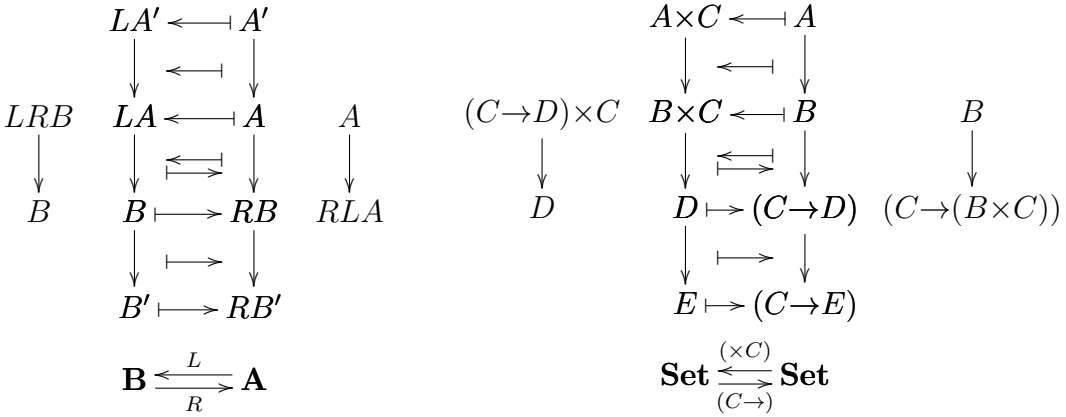
transformations and drawing the internal views of their internal views:





5.5 A way to teach adjunctions

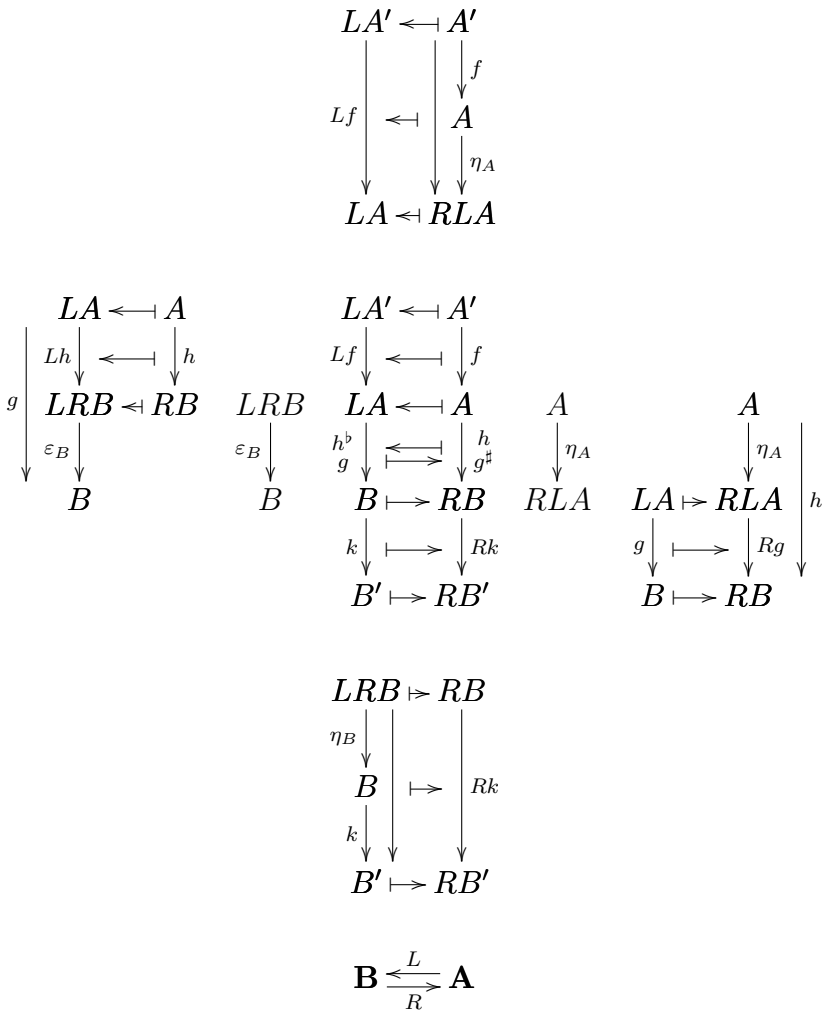
I mentioned in the first sections that I have tested some parts of this language in a seminar course — described here: [Och19] — and that in it I taught Categories starting by adjunctions. Here's how: we started by the basics of λ-calculus and some sections of [PH1], and then I asked the students to define each one of the operations in the right half of the diagram below as λ-terms:



Then we saw the definition of functors, natural transformations and adjunctions, and we checked that the right half is a particular case ("for chil-

dren"!) of the diagram for a generic adjunction in the left half. After that, and after also checking that in the Planar Heyting Algebras of [PH1] we have an adjunction $(\wedge Q) \dashv (Q \to)$, I helped the students to decypher some excerpts of [Awo06].

From the components of the generic adjunction in the diagram above it is possible to build this big diagram:



Let's use these names for its subdiagrams: $\begin{smallmatrix} & & A & & \\ B & C & D & E & F \\ & & G & & \\ & & I & & \end{smallmatrix}$.

A *fully-specified adjunction* between categories $\mathbf{B}$ and $\mathbf{A}$ has lots of components: $(L, R, \varepsilon, \eta, \flat, \sharp, \mathsf{univ}(\varepsilon), \mathsf{univ}(\eta))$, and maybe even others, but usually

we define only some of these components; there is a Big Theorem About Adjunctions (below!) that says how to reconstruct the fully-specified adjunction from some of its components.

Some parts of the diagram above can be interpreted as definitions, like these:

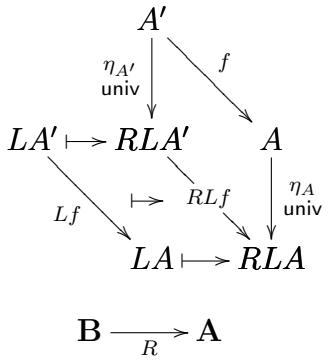
$$Lf := (\eta_A \circ f)^\flat$$
$$g := \varepsilon_B \circ Lh \qquad \varepsilon_B := (\mathrm{id}_{RB})^\flat \qquad \eta_A := (\mathrm{id}_{LA})^\sharp \qquad h := Rg \circ \eta_A$$
$$Rk := (k \circ \eta_B)^\sharp$$

The subdiagrams B and F can also be interpreted in the opposite direction, as:

$$\begin{array}{rcl} g^\sharp & := & (\forall A.\forall g.\exists!h)Ag \\ & = & (\mathsf{univ}_{\varepsilon_B})Ag \end{array} \qquad\qquad \begin{array}{rcl} h^\flat & := & (\forall B.\forall h.\exists!g)Bh \\ & = & (\mathsf{univ}_{\eta_A})Bh \end{array}$$

The notations $(\forall A.\forall g.\exists!h)Ag$ and $(\mathsf{univ}_{\varepsilon_B})Ag$ are clearly abuses of language — but it's not hard to translate them to something formal, and these notations inspired great discussions in the classroom... also, they can help us to understand and formalize constructions like this one,

$$Lf := (\mathsf{univ}_{\eta_A})(LA)(\eta_A \circ f)$$



that are needed in cases like the part (ii) of the Big Theorem.

The Big Theorem About Adjunctions is this — it's the Theorem 2 in [CWM, page 83], but with letters changed to match the ones we are using in our diagrams:

Big Theorem About Adjunctions. Each adjunction $\langle L, R, \sharp \rangle : \mathbf{A} \rightharpoonup \mathbf{B}$ is completely determined by the items in any one of the following lists:

(i) Functors L, R, and a natural transformation $\eta : \mathrm{id}_{\mathbf{A}} \to RL$ such that each $\eta_A : A \to RLA$ is universal to R from A. Then $\sharp$ is defined by (6).

(ii) The functor $R : \mathbf{B} \to \mathbf{A}$ and for each $A \in \mathbf{A}$ an object $L_0 A \in \mathbf{B}$ and a universal arrow $\eta_A : A \to RL_0 A$ from A to R. Then the functor L has object function L_0 and is defined on arrows $f : A' \to A$ by $RLf \circ \eta_{A'} = \eta_A \circ f$.

(iii) Functors L, R, and a natural transformation $\varepsilon : LR \to \mathrm{id}_{\mathbf{B}}$ such that each $\varepsilon_B : LRB \to B$ is universal from L to B. Here $\flat$ is defined by (7).

(iv) The functor $L : \mathbf{A} \to \mathbf{B}$ and for each $B \in \mathbf{B}$ an object $R_0 B \in \mathbf{A}$ and an arrow $\varepsilon_B : LR_0 B \to B$ universal from L to B.

(v) Functors L, R and natural transformations $\eta : \mathrm{id}_{\mathbf{A}} \to RL$ and $\varepsilon : LR \to \mathrm{id}_{\mathbf{B}}$ such that both composites (8) are the identity transformations. Here $\sharp$ is defined by (6) and $\flat$ by (7).

6 Types for Children

We will need a bit of Type Theory in the sections 8.2 and 8.1. We will need some non-standard notational conventions that appear more or less naturally when we present Theory Theory "for children" in the right way — let's see the details.

Section 6 of [Sel13] has a very good presentation of types "for adults": it uses expressions like $A \times B$ and $A \to B$ as and treats them as purely syntactical objects, but each one comes with an "intended meaning". Let's start by defining a version "for children" of that in which these intended meanings become more concrete, and then we will work in the version "for children" and in the version "for adults" in parallel.

6.1 Dependent types

In our version "for children":

- all types are sets,
- some sets are types,
- every finite subset of $\mathbb{N}$ is a type,
- if A and B are types then $A \times B$ and $A \to B$ are types. $A \times B$ is the space of pairs of the form (a, b) in which $a \in A$ and $b \in B$, and $A \to B$ is the space of functions from A to B,
- $a : A$ means $a \in A$ — the distinction between ':' and '$\in$' will only appear in other settings,
- "space of" means "set of". The space of functions from A to B is the set of all functions from A to B, and each function is considered as a set of input-output pairs. So, for example, if $A = \{2, 3\}$ and $B = \{4, 5\}$ then:

$$\begin{array}{rcl} A \times B & = & \{(2,4),(2,5),(3,4),(3,5),\}, \\ A \to B & = & \left\{ \left\{ \begin{smallmatrix}(2,4),\\(3,4)\end{smallmatrix} \right\}, \left\{ \begin{smallmatrix}(2,4),\\(3,5)\end{smallmatrix} \right\}, \left\{ \begin{smallmatrix}(2,5),\\(3,4)\end{smallmatrix} \right\}, \left\{ \begin{smallmatrix}(2,5),\\(3,5)\end{smallmatrix} \right\} \right\} \end{array}$$

- if A is a type and $(C_a)_{a\in A}$ is a family of types indexed by A then $\Pi a{:}A.C_a$ and $\Sigma a{:}A.C_a$ are dependent types defined in the usual way, and $(a{:}A) \to C_a$ and $(a{:}A) \times C_a$ are alternate notations for $\Pi a{:}A.C_a$ and $\Sigma a{:}A.C_a$ (see [NC08, section 2]). Formally,

$$\begin{array}{rcl} \Sigma a{:}A.C_a & = & \{\,(a,c) \in A \times (\bigcup_{a\in A} C_a) \mid a \in A,\ c \in C_a\,\} \\ (a{:}A) \times C_a & = & \{\,(a,c) \in A \times (\bigcup_{a\in A} C_a) \mid a \in A,\ c \in C_a\,\} \\ \Pi a{:}A.C_a & = & \{\,f : A \to (\bigcup_{a\in A} C_a) \mid \forall a \in A.\ f(a) \in C_a\,\} \\ (a{:}A) \to C_a & = & \{\,f : A \to (\bigcup_{a\in A} C_a) \mid \forall a \in A.\ f(a) \in C_a\,\} \end{array}$$

  If $A = \{2,3\}$, $C_2 = \{6,7\}$, and $C_3 = \{7,8\}$ then:

$$\begin{array}{rcl} (a{:}A) \times C_a & = & \{(2,6),(2,7),(3,7),(3,8),\}, \\ (a{:}A) \to C & = & \left\{ \left\{ \begin{smallmatrix}(2,6),\\(3,7)\end{smallmatrix} \right\}, \left\{ \begin{smallmatrix}(2,6),\\(3,8)\end{smallmatrix} \right\}, \left\{ \begin{smallmatrix}(2,7),\\(3,7)\end{smallmatrix} \right\}, \left\{ \begin{smallmatrix}(2,7),\\(3,8)\end{smallmatrix} \right\} \right\}. \end{array}$$

6.2 Witnesses

If P is a proposition we will write $[\![P]\!]$ for its *space of witnesses*, or its *space of proofs*. The exact definition of $[\![P]\!]$ will usually depend on the context, but we always have $[\![P]\!] = \emptyset$ when P is false and $[\![P]\!] \neq \emptyset$ when P is true. In some situations all the witnesses of a proposition P will be identified — this is called *proof irrelevance*; see [NG14, p.340] — and all the spaces of witnesses will be either singletons or empty sets; in other situations some '$[\![P]\!]$'s will have more than one element.

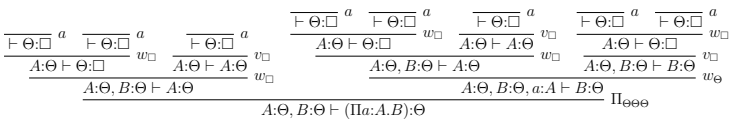
The notation $\langle\!\langle P \rangle\!\rangle$ will denote a witness that P is true. Formally, $\langle\!\langle P \rangle\!\rangle$ is a variable (with a long name!) whose type is $[\![P]\!]$. A good way to remember this notation is that $[\![P]\!]$ looks like a box and $\langle\!\langle P \rangle\!\rangle$ looks like something that comes in that box.

In Agda the operation '$\equiv$' returns a space of proofs of equality. If a and b are expressions with the same type then Agda's '$\mathsf{a} \equiv \mathsf{b}$' corresponds to our $[\![\mathsf{a} = \mathsf{b}]\!]$, and people sometimes use the name '$\mathsf{a{\equiv}b}$' to denote an element of $\mathsf{a} \equiv \mathsf{b}$ — we use $\langle\!\langle \mathsf{a} = \mathsf{b} \rangle\!\rangle$ for that. See the section "Equality" in [WKS20] for simple examples, and Agda's standard library for more examples.

6.3 Judgments

The main objects of Type Theory are *derivable judgements*. A derivable judgment is one that can appear as the root node of a derivation in which each

bar is an application of one the rules in [NG14, p.127]. These derivations are usually huge — for example, here is a derivation for $A{:}\Theta, B{:}\Theta \vdash (\Pi a{:}A.B){:}\Theta$:



so rarely draw them explicitly, and we use other tools to show that certain judgments are derivable.

Every derivable judgment obeys this (taken verbatim from [Sel13, p.52]):

> A typing judgment is an expression of the form
>
> $$x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n \vdash M : A.$$
>
> Its meaning is: "under the assumption that x_i is of type A_i , for $i = 1 \ldots n$, the term M is a well-typed term of type A." The free variables of M must be contained in $x_1, \ldots, x_n$.

Understanding what this means in the version "for children" will take us quite close to understanding that in Type Theory "for adults". We will do that in the next section.

Let me just correct a simplification. The main objects of the Type Theory used in Agda and in most other proof assistants are derivable judgments *with definitions*, as explained in the chapters 8–10 of [NG14]. A judgment with definitions is written as $\Delta; \Gamma \vdash M : N$, where Δ is a list of definitions ([NG14, def.9.2.1]); we will mostly ignore the 'Δ' here.

6.4 Set comprehensions

The part at the left of the '$\vdash$' in a typing judgment is called a *typing context*. Typing contexts also appear in set comprehensions. Let's see an example:

$$\{\, 10a + b \mid a \in \{1,2\}, b \in \{2,3\}, a < b \,\}$$

$$\rightsquigarrow \quad \{\, \underbrace{\underbrace{a \in \{1,2\}}_{\text{generator}}, \underbrace{b \in \{2,3\}}_{\text{generator}}, \underbrace{a < b}_{\text{filter}}}_{\text{context}} \;;\; \underbrace{10a + b}_{\text{expression}} \,\}$$

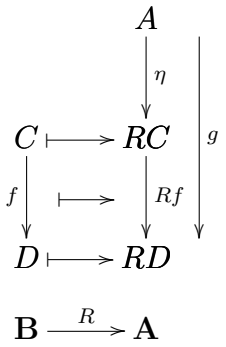
We rewrote the comprehension $\{\ \langle \mathsf{expr} \rangle \mid \langle \mathsf{context} \rangle\ \}$ to $\{\ \langle \mathsf{context} \rangle\ ;\ \langle \mathsf{expr} \rangle\ \}$ for clarity, and we marked which parts of the context act as "generators" and which ones act as "filters". The context above can be rewritten in type-theoretical notation as:

$$a : \{1, 2\},\ b : \{2, 3\},\ \langle\!\langle a{<}b \rangle\!\rangle : [\![a{<}b]\!]$$

A *value* for that context is a triple $(a, b, \langle\!\langle a{<}b \rangle\!\rangle)$, where $a \in \{1, 2\}$, $b \in \{2, 3\}$, and $\langle\!\langle a{<}b \rangle\!\rangle$ is a guarantee that $a < b$ is true.

6.5 Omitting types

The diagram at the left below is a copy of the one from sec.4.2, but now we will interpret it in a different way, as a "dictionary of (default) types". For example, it says that when the symbol η appears without a type its type will be the default one given by the diagram: $\eta : A \to RC$, or $\eta : \mathbf{A}(A, RC)$. The default types are listed at the right.



$$\begin{array}{lcl} \mathbf{A} & \text{is} & \text{a category} \\ \mathbf{B} & \text{is} & \text{a category} \\ R & : & \mathbf{A} \to \mathbf{B} \\ A & \in & \mathbf{A} \\ C & \in & \mathbf{B} \\ D & \in & \mathbf{B} \\ \eta & : & A \to RC \\ f & : & C \to D \\ g & : & A \to RD \end{array}$$

6.6 Indefinite articles

We will use the diagram above to redefine universalness. In our old definition, from sec.4.2, universalness is just a "property"; in our new definition it will be a pair made of a "structure" and a "property" (see sec.8.2).

Suppose that

$$\begin{array}{lcl} \mathbf{A} & \text{is} & \text{a category}, \\ \mathbf{B} & \text{is} & \text{a category}, \\ R & : & \mathbf{A} \to \mathbf{B}, \\ A & \in & \mathbf{A}, \\ C & \in & \mathbf{B}, \\ \eta & : & A \to RC, \end{array}$$

and let $\sharp$ be this operation:

$$\sharp \quad = \quad \lambda D.\lambda f.\, Rf \circ \eta.$$

Note that the types of D and f are given by the diagram: D is an object of $\mathbf{B}$, and $f : C \to D$. Then "universalness" for the tuple $(\mathbf{A}, \mathbf{B}, R, A, C, \eta)$ is a pair $(\flat, \langle\!\langle \flat = \sharp^{-1} \rangle\!\rangle)$, in which $\flat$ is an operation "that for each D takes each g to an f" and $(\flat = \sharp^{-1})$ is a shorthand for this proposition:

$$\begin{array}{l} (\forall D.\forall f.\, \flat\, D\, (\sharp\, D\, f) = f) \quad \wedge \\ (\forall D.\forall g.\, \sharp\, D\, (\flat\, D\, g) = g). \end{array}$$

The component $\langle\!\langle \flat = \sharp^{-1} \rangle\!\rangle$ of the universalness is a witness that guarantess that this proposition holds.

The types of $\flat$ and $\langle\!\langle \flat = \sharp^{-1} \rangle\!\rangle$ are:

$$\begin{array}{rcl} \flat & : & (D : \mathrm{Objs}_{\mathbf{B}}) \to (\mathbf{A}(A, RD) \to \mathbf{B}(C, D)) \\ \langle\!\langle \flat = \sharp^{-1} \rangle\!\rangle & : & [\![\, (\forall D.\forall f.\, \flat\, D\, (\sharp\, D\, f) = f) \ \wedge \\ & & \quad (\forall D.\forall g.\, \sharp\, D\, (\flat\, D\, g) = g) \,]\!] \end{array}$$

We can use a trick with indefinite articles to obtain the type of $\flat$. Let's overload the notations $[\![\cdot]\!]$ and $\langle\!\langle \cdot \rangle\!\rangle$: with their new meanings '$[\![\alpha]\!]$' will be pronounced "the type of α", and '$\langle\!\langle \alpha \rangle\!\rangle$' will be "an α", "an object with the same type of α", or "an element of $[\![\alpha]\!]$". Then

$\flat$ is an operation that for each D takes each g to an f

becomes:

$$\flat \quad = \quad \lambda D.\lambda g.\langle\!\langle f \rangle\!\rangle$$

The indefinite article in this $\langle\!\langle f \rangle\!\rangle$ is contagious: we read the equality above not as "$\flat$ is *the* operation that takes each D..." but as "$\flat$ is *an* operation that...". We don't know the value of $\lambda D.\lambda g.\langle\!\langle f \rangle\!\rangle$ but we can calculate its type:

$$\begin{array}{rcl} f & : & C \to D \\ [\![f]\!] & : & \mathbf{B}(C, D) \\ \langle\!\langle f \rangle\!\rangle & : & \mathbf{B}(C, D) \\ g & : & A \to RD \\ [\![g]\!] & : & \mathbf{A}(A, RD) \\ D & : & \mathrm{Objs}_{\mathbf{B}} \\ [\![\lambda D.\lambda g.\langle\!\langle f \rangle\!\rangle]\!] & : & \mathrm{Objs}_{\mathbf{B}} \to (\mathbf{A}(A, RD) \to \mathbf{B}(C, D)) \end{array}$$

We will see how to represent universalness in diagrams in sec.8.2.

6.7 "Physicists' notation"

Books like [TG88] use a notation in which expressions like "$z = z(x, y(x))$" are allowed — the same symbol can be used both as a (dependent!) variable and as the name of a function, and arguments can be omitted — and in a context in which $y = y(x)$ the default meaning for y_0 is $y(x_0)$. In many areas of Mathematics that notation has been phased out (see [JIB22, sec.3.3]) and replaced by one in which the names of the bound variables matter very little.

Let's call the older notation "physicist's notation" and the newer one "mathematician's notation"; these names are not standard at all. If we apply the ideas of the "physicist's notation" to judgments we can abbreviate

$$a : A, b : B(a), c : C(a, b) \vdash d(a, b, c) : D(a, b, c)$$

to just $a : A, b : B, c : C \vdash d : D$, or even to $a, b, c \vdash d$. Some of the conventions of DL were inspired by conventions from "physicist's notation".

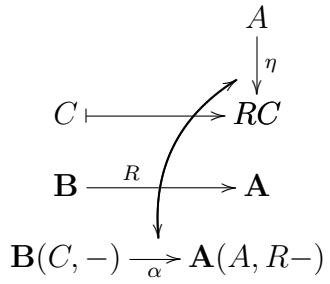
7 The Basic Example as a skeleton

In the sections 2 and 3 I claimed that the diagram of the Basic Example is a "skeleton" of a certain theorem, in the sense that both the statement and the proof of that theorem can be reconstructed from just the diagram and very few extra hints. Let's see the details of this.

7.1 Reconstructing its functors

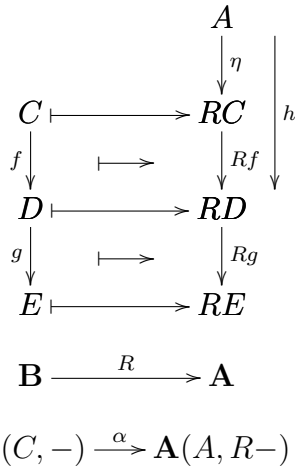
Let's call this diagram — the diagram of the Basic Example — $\mathsf{Y0}$:

$\mathsf{Y0}$:=



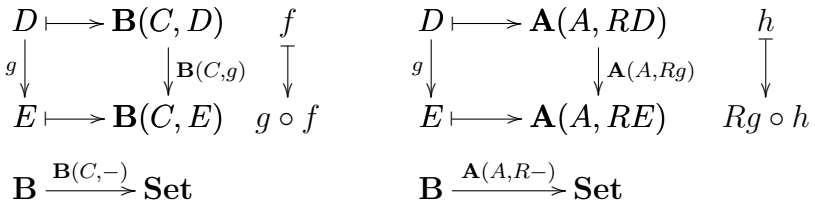
We don't know yet the precise meaning of the functors $\mathbf{B}(C,-)$ and $\mathbf{A}(A,R-)$, but if we enlarge $\mathsf{Y0}$ to this — note that we are omitting the curved bijection by convenience,

$\mathsf{Y0}^{+}$:=



and we draw the internal views of $\mathbf{B}(C,-)$ and $\mathbf{A}(A,R-)$, then the meanings

for $\mathbf{B}(C,-)$ and $\mathbf{A}(A,R-)$ become obvious:



So:

$$\begin{array}{rcl}
\mathbf{B}(C,-) & : & \mathbf{B} \to \mathbf{Set} \\
\mathbf{B}(C,-)_0 & := & \lambda D.\mathbf{B}(C,D) \\
\mathbf{B}(C,-)_1 & := & \lambda g.\lambda f.g \circ f \\
\mathbf{A}(A,R-) & : & \mathbf{B} \to \mathbf{Set} \\
\mathbf{A}(A,R-)_0 & := & \lambda D.\mathbf{A}(A,RD) \\
\mathbf{A}(A,R-)_1 & := & \lambda g.\lambda h.Rg \circ h
\end{array}$$

7.2 Natural transformations

In sec.5.3 we saw that a natural transformation is a pair. An NT $\alpha : \mathbf{B}(C,-) \to \mathbf{A}(A,R-)$ is a pair $(\alpha_0, \mathsf{sqcond}_\alpha)$, where sqcond_α is this:

$$\begin{array}{rcl}
\mathsf{sqcond}_\alpha & = & \forall D.\forall E.\forall g.\forall f.\ (\mathbf{A}(A,Rf) \circ \alpha D) = (\alpha E \circ \mathbf{B}(C,f)) \\
& = & \forall D.\forall E.\forall g.\forall f.\ (Rg \circ (\alpha D f)) = (\alpha E(g \circ f)) \\
& = & \forall D.\forall E.\forall g.\forall f.\ (Rg \circ (\alpha_0 D f)) = (\alpha_0 E(g \circ f))
\end{array}$$

We can visualize what this "means" using the two diagrams at the top in the next page.

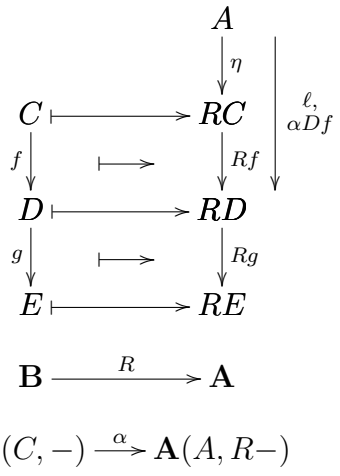
Suppose that we define a natural transformation α_η by saying that $(\alpha_\eta)_0 = \lambda D.\lambda f.Rf \circ \eta$. Then we can either affirm that $\mathsf{sqcond}_{\alpha_\eta}$ "is obvious" or verify that it holds. Substituing α_0 by $\lambda D.\lambda f.Rf \circ \eta$ we obtain:

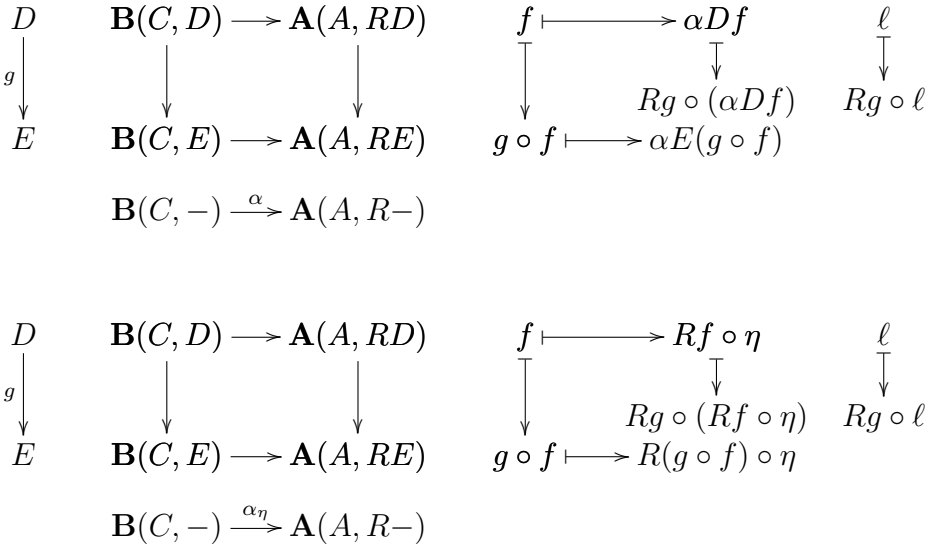
$$\mathsf{sqcond}_{\alpha_\eta} \quad = \quad \forall D.\forall E.\forall g.\forall f.\ (Rg \circ (Rf \circ \eta)) = (R(g \circ f) \circ \eta)$$

which is clearly true, so $\mathsf{sqcond}_{\alpha_\eta}$ holds, and α_η is a natural transformation for every $\eta : A \to RC$. We can define an operation $(\eta \to \alpha)$ by:

$$(\eta \mapsto \alpha) \quad := \quad \lambda\eta.((\alpha_\eta)_0, \mathsf{sqcond}_{\alpha_\eta})$$

Without abbreviations this definition would be very big. The lower third of the diagram in the next page shows how visualize what $\mathsf{sqcond}_{\alpha_\eta}$ means.


A
η
C ⟼ RC
ℓ, αDf
f ⟼ Rf
D ⟼ RD
g ⟼ Rg
E ⟼ RE
B —R→ A
B(C, −) —α→ A(A, R−)


D
g
E
B(C, D) → A(A, RD)
B(C, E) → A(A, RE)
B(C, −) —α→ A(A, R−)
f ⟼ αDf
Rg ∘ (αDf)
g ∘ f ⟼ αE(g ∘ f)
ℓ
Rg ∘ ℓ

D
g
E
B(C, D) → A(A, RD)
B(C, E) → A(A, RE)
B(C, −) —αη→ A(A, R−)
f ⟼ Rf ∘ η
Rg ∘ (Rf ∘ η)
g ∘ f ⟼ R(g ∘ f) ∘ η
ℓ
Rg ∘ ℓ

We can define an operation $(\alpha \mapsto \eta)$ by:

$$\begin{array}{lcl} (\alpha \mapsto \eta) & := & \lambda\alpha.\, \alpha_0\, C\, \mathrm{id}_C, \\ (\eta \mapsto \alpha_0) & := & \lambda\eta.\lambda D.\lambda f.\, Rf \circ \eta, \\ (\eta \mapsto \alpha) & := & \lambda\eta.((\eta \mapsto \alpha_0)(\eta),\ \mathsf{sqcond}_{(\text{something})}). \end{array}$$

We can prove that the operations $(\eta \mapsto \alpha)$ and $(\alpha \mapsto \eta)$ are mutually inverse, but this is tricky. The proof contains a step that is hard to visualize, and that is often stated as a slogan, like this (from [Leinster, p.97] and [CWM, p.61]):

> A natural transformation $\alpha : \mathbf{B}(C, -) \to \mathbf{A}(A, R-)$
> is determined by its value at id_C.

The proof of that step requires instantiating sqcond_α, i.e.,

$$\forall D.\forall E.\forall g : D \to E.\ \forall f : C \to D.\ (Rg \circ (\alpha_0 Df)) = (\alpha_0 E(g \circ f))$$

at $D := C$, $E := D$, $g := f$, and $f := \mathrm{id}_C$. If we do this in two sub-steps — first $D := C$ and $E := D$, and then $g := f$ and $f := \mathrm{id}_C$ — we see that after the first sub-step we get this:

$$\forall g : C \to D.\ \forall f : C \to C.\ (Rg \circ (\alpha_0 Cf)) = (\alpha_0 D(g \circ f))$$

The variables g and f have sort of changed their types, and some people (like me!) would prefer to rename them, to, say:

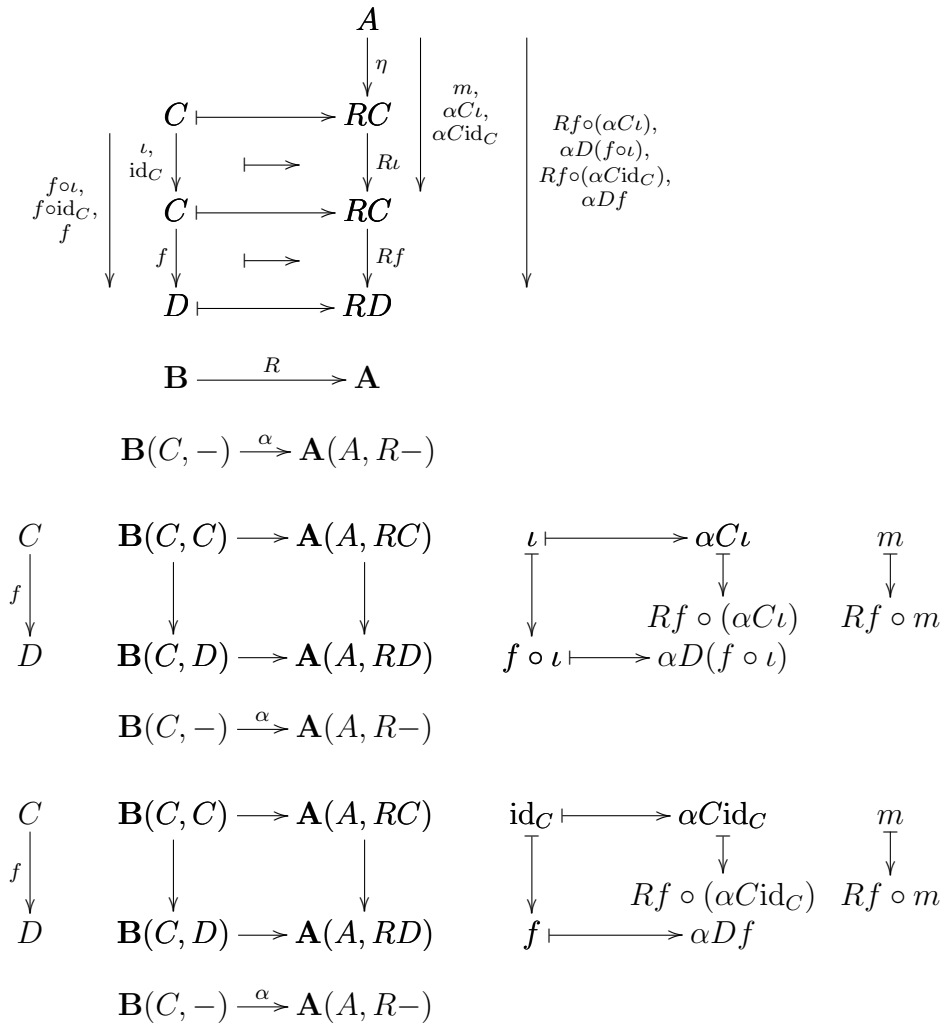
$$\forall f : C \to D.\ \forall \iota : C \to C.\ (Rf \circ (\alpha_0 C\iota)) = (\alpha_0 D(f \circ \iota))$$

The diagrams in the next page show the renamed version.

To prove that our operations $(\alpha \mapsto \eta)$ and $(\eta \mapsto \alpha)$ are mutually inverse we need to prove that the round trips $(\alpha \mapsto \eta \mapsto \alpha)$ and $(\eta \mapsto \alpha \mapsto \eta)$ are both identities. To prove that $(\alpha \mapsto \eta \mapsto \alpha) = \mathrm{id}$, let's define $\eta_\alpha := (\alpha \to \eta)(\alpha)$ and $\alpha_{\eta_\alpha} := (\eta \mapsto \alpha)(\eta_\alpha)$ and . The proof of $(\alpha \mapsto \eta \mapsto \alpha) = \mathrm{id}$ includes this sequence of equalities:

$$\begin{array}{rcl} (\alpha_{\eta_\alpha})_0 Df & = & (\eta \mapsto \alpha_0)((\alpha \mapsto \eta)(\alpha))Df \\ & = & (\eta \mapsto \alpha_0)(\alpha C \mathrm{id}_C)Df \\ & = & (\lambda\eta.\lambda D.\lambda f.\ Rf \circ \eta)(\alpha C \mathrm{id}_C)Df \\ & = & (\lambda D.\lambda f.\ Rf \circ (\alpha C \mathrm{id}_C))Df \\ & = & Rf \circ (\alpha C \mathrm{id}_C) \\ & = & \alpha Df \end{array}$$

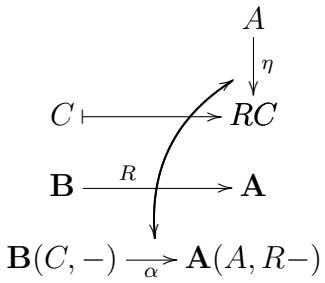
that uses our hard step in its last equality. The details, including the proof of $(\eta \mapsto \alpha \mapsto \eta) = \mathrm{id}$, can be found in [Och22].



7.3 The full reconstruction

We have just reconstructed all the typings and definitions for the diagram Y0. Here is the full reconstruction, except for the "proof terms" like respids, assoc, idL and idR for each functor, sqcond for each natural transformations,

and the proofs that both round trips in the bijections are identity maps:



$\mathbf{A}$ is a category,
$\mathbf{B}$ is a category,
$R : \mathbf{B} \to \mathbf{A}$,
$A \in \mathbf{A}$,
$C \in \mathbf{B}$,
$\eta : A \to RD$,

$\mathbf{B}(C, -) : \mathbf{B} \to \mathbf{Set}$,
$\mathbf{B}(C, -)_0 := \lambda D.\mathbf{B}(C, D)$,
$\mathbf{B}(C, -)_1 := \lambda g.\lambda f.\ g \circ f$,

$\mathbf{A}(A, R-) : \mathbf{A} \to \mathbf{Set}$,
$\mathbf{A}(A, R-)_0 := \lambda D.\mathbf{A}(A, RD)$,
$\mathbf{A}(A, R-)_1 := \lambda g.\lambda h.Rg \circ h$,

$\alpha : \mathbf{B}(C, -) \to \mathbf{A}(A, R-)$,

$(\eta \mapsto \alpha_0) := \lambda \eta.\lambda D.\lambda f.\ Rf \circ \eta$,
$(\alpha \mapsto \eta) := \lambda \alpha.\ \alpha C(\mathrm{id}_C)$,

or:
$\alpha_0 := \lambda D.\lambda f.Rf \circ \eta$,
$\eta := \alpha C(\mathrm{id}_C)$.

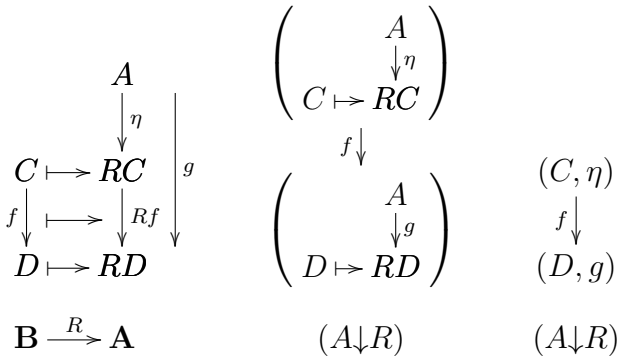
It is quite short — *if we treat the proof terms as "obvious".*

8 Extensions to the diagrammatic language

Our diagrammatic language and the list of conventions in Section 2 can be extended — "by the user" — in zillions of ways. Let's see some examples of extensions.

8.1 A way to define new categories

We saw in the sections 5.2 and 7.1 how to use diagrams to define functors, and in sections 5.3 and 7.2 how to define natural transformations. We can define new categories by diagrams, too.



My favorite way — a syntax sugar! — of visualizing the comma category $(A{\downarrow}R)$ is the middle third of the diagram above, in which the objects of $(A{\downarrow}R)$ are depicted as L-shaped diagrams. To understand the typings and the commutativity conditions we have to look at the left third — it indicates that f must obey $Rf \circ \eta = g$. The right third shows a generic morphism in $(A{\downarrow}R)$ without the syntax sugar, but we still have to look at the left third

to type it. We have:

In a context in which	$\mathbf{A}$ is a category,
	$\mathbf{B}$ is a category,
	$R : \mathbf{B} \to \mathbf{A}$,
	A is an object of $\mathbf{A}$,
we define the category	$(A{\downarrow}R)$ as follows:
An object of	$(A{\downarrow}R)$
is a pair	(C, η)
in which	$C : \mathbf{B}_0$
and	$\eta : \mathrm{Hom}_{\mathbf{A}}(A, RC)$;
so	$(C, \eta) : (C{:}\mathbf{B}_0) \times \mathrm{Hom}_{\mathbf{A}}(A, RC)$
and	$(A{\downarrow}R)_0 := (C{:}\mathbf{B}_0) \times \mathrm{Hom}_{\mathbf{A}}(A, RC)$.
A morphism	$f : (C, \eta) \to (D, g)$ in $(A{\downarrow}R)$
is an	$f : \mathrm{Hom}_{\mathbf{B}}(C, D)$ such that $Rf \circ \eta = g$,
or equivalently a pair	$(f, \langle\!\langle Rf \circ \eta = g \rangle\!\rangle)$;
we have	$(f, \langle\!\langle Rf \circ \eta = g \rangle\!\rangle) : (f{:}\mathrm{Hom}_{\mathbf{B}}(C, D)) \times [\![Rf \circ \eta = g]\!]$,
so	$\mathrm{Hom}_{(A{\downarrow}R)}((C, \eta), (D, g)) :=$
	$(f{:}\mathrm{Hom}_{\mathbf{B}}(C, D)) \times [\![Rf \circ \eta = g]\!]$.

This defines formally the first two components of the category $(A{\downarrow}R)$. Remember that a category $\mathbf{C}$ has seven components:

$$\mathbf{C} = (\mathbf{C}_0, \mathrm{Hom}_{\mathbf{C}}, \mathrm{id}_{\mathbf{C}}, \circ_{\mathbf{C}}; \mathsf{assoc}_{\mathbf{C}}, \mathsf{idL}_{\mathbf{C}}, \mathsf{idR}_{\mathbf{C}})$$

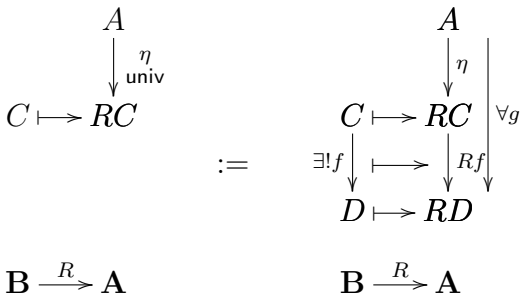
We are pretending that the other components of $(A{\downarrow}R)$ are "obvious" in the sense of Section 3. Note the we used the notations for dependent types and witnesses of the sections 6.1 and 6.2.

8.2 Universalness as something extra

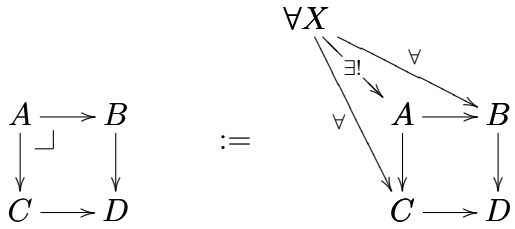
We can consider that an universal arrow is an arrow $\eta : A \to RC$ with something extra. We saw how to represent this "something extra" in Type Theory: a universal arrow η is a pair $(\eta, \mathsf{univ}_\eta)$, where univ_η is its "universalness", that we defined in one way in sec.4.2 and in another way in sec.6.6.

Universalness is just one '-ness' among many. Several of these "-ness"es have standard graphical representations: for example pullbackness is indicated by a '⌟', and monicness is indicated by a tail like this: '$\rightarrowtail$'. [FS90]

defines lots of graphical representations for "-ness"es starting on its page 37. We will sometimes use an ':=' to define a new annotation that is an abbreviation for extra structure:



This is pullbackness:

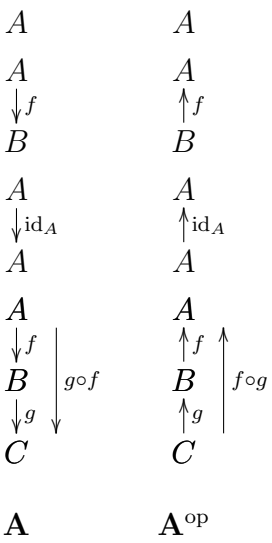


8.3 Opposite categories

Suppose that we have a diagram $A \xrightarrow{f} B \xrightarrow{g} C$ in a category $\mathbf{A}$. There are several different notations for the corresponding diagram in $\mathbf{A}^{\mathrm{op}}$: for example, in [CWM, p.33] it would be written as $A \xleftarrow{f^{\mathrm{op}}} B \xleftarrow{g^{\mathrm{op}}} C$, while in [AT11, p.15] as $A \xleftarrow{f} B \xleftarrow{g} C$. The convention (COT) says that the notation in our diagrams should be as close as possible to the notation in the original text — so let's see how to support the notation in [AT11], that looks a bit harder than the one in [CWM].

We want to define a new category, $\mathbf{A}^{\mathrm{op}}$, using tricks similar to the ones in Section 8.1, but now we can't pretend that the new composition is obvious. We will define $(\mathbf{A}^{\mathrm{op}})_0$, $\mathrm{Hom}_{\mathbf{A}^{\mathrm{op}}}$, $\mathrm{id}_{\mathbf{A}^{\mathrm{op}}}$, and $\circ_{\mathbf{A}^{\mathrm{op}}}$ without any textual explanations, with just the diagrams to convince the readed that our definitions

are reasonable.



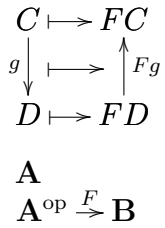
$$\mathbf{A}_0 =: (\mathbf{A}^{\mathrm{op}})_0$$

$$\mathrm{Hom}_{\mathbf{A}}(A, B) =: \mathrm{Hom}_{\mathbf{A}^{\mathrm{op}}}(B, A)$$

$$\mathrm{id}_{\mathbf{A}}(A) =: \mathrm{id}_{\mathbf{A}^{\mathrm{op}}}(A)$$

$$g \circ_{\mathbf{A}} f =: f \circ_{\mathbf{A}^{\mathrm{op}}} g$$

In the diagram below $F : \mathbf{A}^{\mathrm{op}} \to \mathbf{B}$ is a contravariant functor, and the $\mathbf{A}$ above $\mathbf{A}^{\mathrm{op}}$ indicates that $g : C \to D$ is a morphism of $\mathbf{A}$, not of $\mathbf{A}^{\mathrm{op}}$. I am not very happy with this trick but I haven't found a better alternative yet.

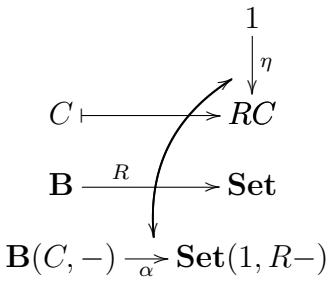


8.4 The Yoneda Lemma

The formalization of Y0 as a series of typings and definitions in Section 7.3 *suggests* that *some* operations from Type Theory that can be applied on the formalization side should be translatable to the diagram side; for example, substitution. This one clearly works: if we substitute $\mathbf{A}$ by **Set** and A by

the set 1 we get this,

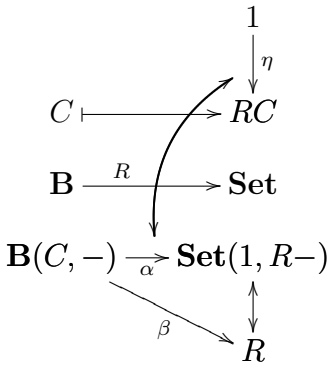
$$\mathsf{Y0}\begin{bmatrix}\mathbf{A} := \mathbf{Set} \\ A := 1\end{bmatrix} \quad = \quad$$



For each object S of **Set** we have a bijection between elements of S and morphisms $1 \to S$. We will denote the morphism from 1 to S that "chooses" an element $s \in S$ by $\ulcorner s \urcorner$; the pronounciation of $\ulcorner s \urcorner$ is "the name of s". We have a bijection between 's's and $\ulcorner s \urcorner$s:

For each $D \in \mathbf{B}$ we have a bijection $\mathbf{Set}(1, RD) \leftrightarrow RD$ — and we can use these bijections to build a natural isomorphism $\mathbf{Set}(1, R-) \leftrightarrow R$. We will draw it vertically, and complete the triangle:

$$\mathsf{Y1} \quad := \quad$$



We can use that natural isomorphism to obtain 'β's from 'α's, and vice-versa, by composition. We could draw an arrow for the bijection between 'α's and 'β's and another arrow for the bijection between 'η's and elements of RC, but we will prefer to omit them.

We will call the diagram above Y1. It doesn't have a single top-level arrow, so we can't apply the convention (CTL) to it, and we need to specify its "meaning" explicitly. We will consider that its three bijections are top-

level objects, and so the diagram Y1 says that we have these bijections:

$$\begin{array}{l} RC \\ \leftrightarrow\ \mathbf{Set}(1, RC) \\ \leftrightarrow\ \mathbf{Set}^{\mathbf{B}}(\mathbf{B}(C,-), \mathbf{Set}(1, R-)) \\ \leftrightarrow\ \mathbf{Set}^{\mathbf{B}}(\mathbf{B}(C,-), R) \end{array}$$

The Yoneda Lemma "is" the bijection $RC \leftrightarrow \mathbf{Set}^{\mathbf{B}}(\mathbf{B}(C,-), R)$ — check how it is defined in [Leinster, thm.4.2.1], [Riehl, thm.2.2.4], [CWM, p.61], [Awo06, lemma 8.2]. Some books show how to calculate the element of RC associated to a given β and vice-versa, and most treat $\mathbf{Set}(1, R-)$ as something secondary. If we represent this idea of the Yoneda Lemma in the same format the we used in sec.7.3, we get this:

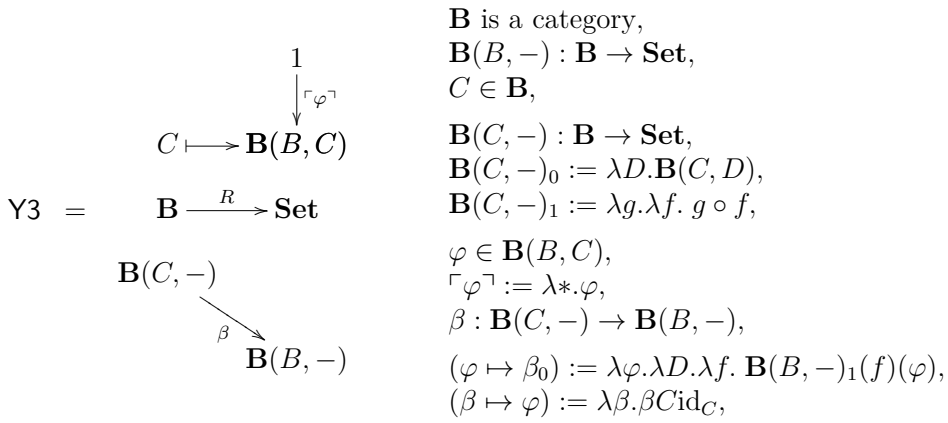
$$\mathsf{Y2} \quad := \quad \begin{array}{ccc} & & 1 \\ & & \downarrow{\scriptstyle \ulcorner e \urcorner} \\ C & \longmapsto & RC \\ \mathbf{B} & \xrightarrow{R} & \mathbf{Set} \\ \mathbf{B}(C,-) & \searrow^{\beta} & R \end{array} \quad \begin{array}{l} \mathbf{B} \text{ is a category,} \\ R : \mathbf{B} \to \mathbf{Set}, \\ C \in \mathbf{B}, \\ \mathbf{B}(C,-) : \mathbf{B} \to \mathbf{Set}, \\ \mathbf{B}(C,-)_0 := \lambda D.\mathbf{B}(C,D), \\ \mathbf{B}(C,-)_1 := \lambda g.\lambda f.\ g \circ f, \\ e \in RC, \\ \ulcorner e \urcorner := \lambda *.e, \\ \beta : \mathbf{B}(C,-) \to R, \\ (e \mapsto \beta_0) := \lambda e.\lambda D.\lambda f.\ Rf(e), \\ (\beta \mapsto e) := \lambda \beta.\beta C \mathrm{id}_C, \end{array}$$

8.5 The Yoneda embedding

Let's define Y3 as the result of this substituion:

$$\mathsf{Y3} \quad = \quad \mathsf{Y2} \begin{bmatrix} R := \mathbf{B}(B,-) \\ e := \varphi \end{bmatrix}$$

We have:



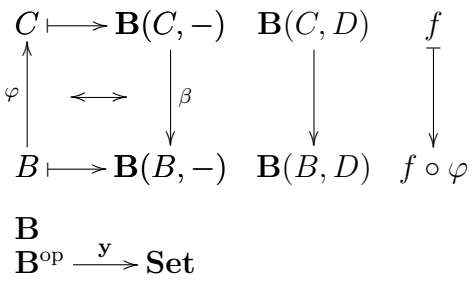
B is a category,
$\mathbf{B}(B,-) : \mathbf{B} \to \mathbf{Set}$,
$C \in \mathbf{B}$,

$\mathbf{B}(C,-) : \mathbf{B} \to \mathbf{Set}$,
$\mathbf{B}(C,-)_0 := \lambda D.\mathbf{B}(C,D)$,
$\mathbf{B}(C,-)_1 := \lambda g.\lambda f.\ g \circ f$,

$\varphi \in \mathbf{B}(B,C)$,
$\ulcorner \varphi \urcorner := \lambda *.\varphi$,
$\beta : \mathbf{B}(C,-) \to \mathbf{B}(B,-)$,

$(\varphi \mapsto \beta_0) := \lambda\varphi.\lambda D.\lambda f.\ \mathbf{B}(B,-)_1(f)(\varphi)$,
$(\beta \mapsto \varphi) := \lambda\beta.\beta C \mathrm{id}_C$,

The formulas in $(\varphi \mapsto \beta_0)$ and $(\beta \mapsto \varphi)$ can be simplified, and we can derive this other diagram from it:



B is a category,
$B \in \mathbf{B}$,
$C \in \mathbf{B}$,

$\varphi : B \to C$,
$\beta : \mathbf{B}(C,-) \to \mathbf{B}(B,-)$,
$(\varphi \mapsto \beta_0) := \lambda\varphi.\lambda D.\lambda f.\ f \circ \varphi$,
$(\beta \mapsto \varphi) := \lambda\beta.\beta C \mathrm{id}_C$,

Let's call it Y4. This is one of the *Yoneda Embeddings*; compare that diagram with the ones in [Riehl, p.60]. In [Och22] we show how to formalize it in Agda as a corollary of Y0, Y1, Y2, and Y3.

Note that the functor $C \mapsto \mathbf{B}(C,-)$ in Y4 is contravariant, and we used the trick from sec.8.3 to indicate this.

8.6 Representable functors

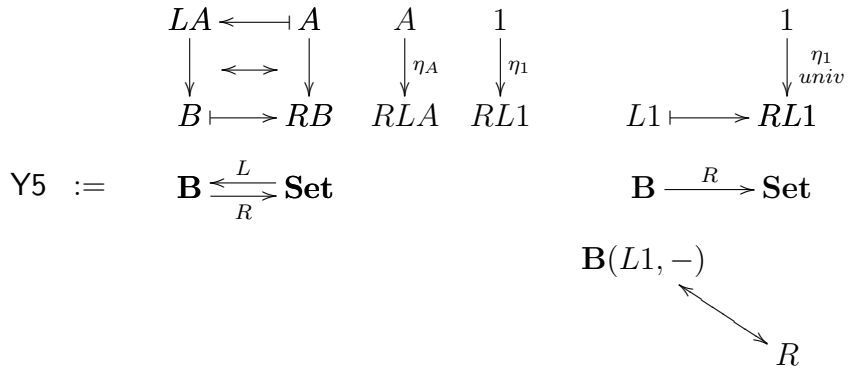
Some books, like [Leinster] and [Riehl], present representable functors first, then lots of examples, and only then they present the Yoneda Lemma. In our diagram Y1 a *representation* for the functor R is a pair (C, β) such that the natural transformation β is a natural iso. It is easy to see that we

have these bijections, or bi-implications:

$$\begin{array}{l} \text{natural iso-ness of } \beta \\ \leftrightarrow \text{ natural iso-ness of } \alpha \\ \leftrightarrow \text{ universalness of } \eta \end{array}$$

The last one holds in the diagram Y0 too.

Many of the textbook examples of representable functors are consequences of a simple theorem that shows that every functor $R : \mathbf{B} \to \mathbf{Set}$ with a left adjoint is representable. The diagram Y5 below is a skeleton for a proof of that theorem:



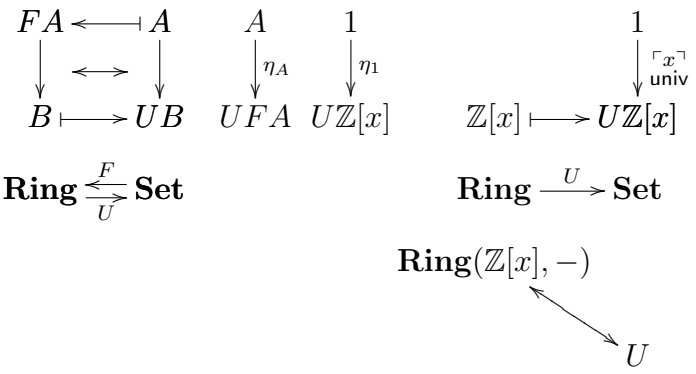
The unit arrow $\eta_1 : 1 \to RL1$ is universal — note the 'univ' in the upper right arrow — and so its associated 'β' is a natural iso; we drew it with a '$\leftrightarrow$'.

Statements like "the functor *blah* is representable" are very common in CT texts, and their natural isos is usually written just like "$R \cong \mathbf{B}(L1, -)$", without a name. A good way to draw the missing diagrams for a text should not force us to invent names, and so we should be allowed to omit the names of arrows when this is convenient.

This is the example (iv) in [Riehl, p.52]:

> (iv) The functor U : **Ring** $\to$ **Set** is represented by the unital ring $\mathbb{Z}[x]$, the polynomial ring in one variable with integer coefficients. A unital ring homomorphism $Z[x] \to R$ is uniquely determined by the image of x; put another way, $\mathbb{Z}[x]$ is the *free unital ring on a single generator*.

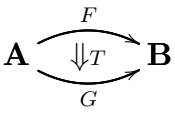
The diagram below is a good way to visualize that:



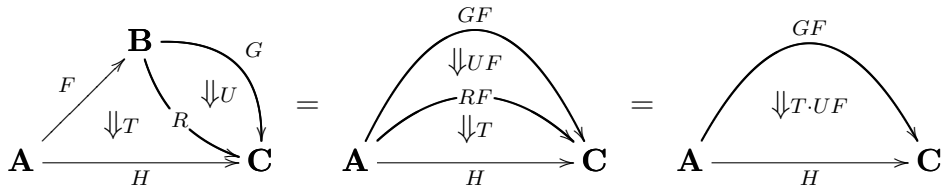
Its left half is useful for when we want to remember how that representation is generated by an adjunction of the form $F \dashv U$, but it can be omitted. In sec.2 I drew that diagram without its left half.

8.7 The 2-category of categories

Natural transformations are often drawn as '⇒'s in the middle of "cells" whose walls are functors. If $F, G : \mathbf{A} \to \mathbf{B}$ are functors and $T : F \to G$ is natural transformation, then $\mathbf{A}, \mathbf{B}, F, G, T$ are drawn like this:



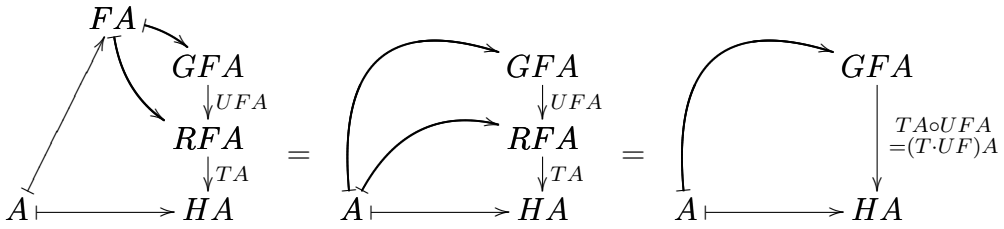
There are several ways to compose functors and natural transformations — see [Riehl, section 1.7], [Haz19, section A.5], and [Pow90] for the details and the precise terminology. For example, in



we used "whiskering" and then "vertical composition".

We can use internal views to lower the level of abstraction of the diagrams above. If we draw the images of an object $A \in \mathbf{A}$ by the functors and natural

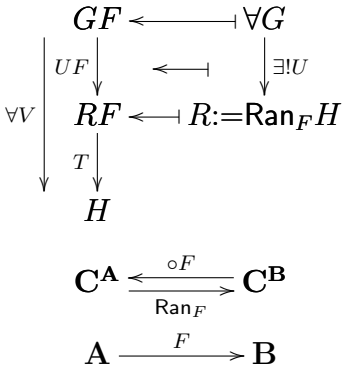
transformations we get:

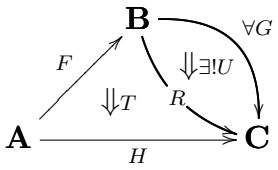


Note that this process of taking a "pasting diagram" and looking at its internal view — in which many names become longer and some nodes are duplicated — is the opposite of what people usually do in the literature: they usually go from pasting diagrams to string diagrams, in which most names are omitted. See [Mar14] and [Haz19, section A.5].

8.8 Kan extensions

Kan extensions are usually drawn using 2-cells ([Riehl, definition 6.1.1]), but they can also be drawn as adjunctions ([Riehl, proposition 6.1.5], [CWM, section X.3]). Let's see how to draw them in both ways at the same time in a way that makes the translation clear. Here is the diagram:





We will consider right Kan extensions only.

Fix $F : \mathbf{A} \to \mathbf{B}$ and a category $\mathbf{C}$. We have a functor $(\circ F) : \mathbf{C}^{\mathbf{B}} \to \mathbf{C}^{\mathbf{A}}$. Suppose that it has a right adjoint, $(\circ F) \dashv \mathsf{Ran}_F$. For each natural transformation $H : \mathbf{A} \to \mathbf{C}$ its image by Ran_f, $R := \mathsf{Ran}_F H$, is a natural transformation $R : \mathbf{B} \to \mathbf{C}$. We have:

$$\begin{aligned}
(\circ F) &\dashv \mathsf{Ran}_H \\
\mathrm{Hom}_{\mathbf{C}^{\mathbf{A}}}((\circ F)-,-) &\cong \mathrm{Hom}_{\mathbf{C}^{\mathbf{B}}}(-,\mathsf{Ran}_F-), \\
\mathrm{Hom}_{\mathbf{C}^{\mathbf{A}}}((\circ F)G,H) &\cong \mathrm{Hom}_{\mathbf{C}^{\mathbf{B}}}(G,\mathsf{Ran}_F H), \\
\mathrm{Hom}_{\mathbf{C}^{\mathbf{A}}}(G\circ F,H) &\cong \mathrm{Hom}_{\mathbf{C}^{\mathbf{B}}}(G,\mathsf{Ran}_F H), \\
\mathrm{Hom}_{\mathbf{C}^{\mathbf{A}}}(GF,H) &\cong \mathrm{Hom}_{\mathbf{C}^{\mathbf{B}}}(G,R)
\end{aligned}$$

and if we substitute $[G := R]$ in $\mathrm{Hom}_{\mathbf{C}^{\mathbf{B}}}(G,R)$ and we transpose id_R to the left we obtain a morphism $T : RF \to H$. The pair (R,H) obeys a certain universal property, that we will call "Ran-ness":

$$\forall G.\ \forall V.\ \exists! U.\ (T \cdot UF) = V.$$

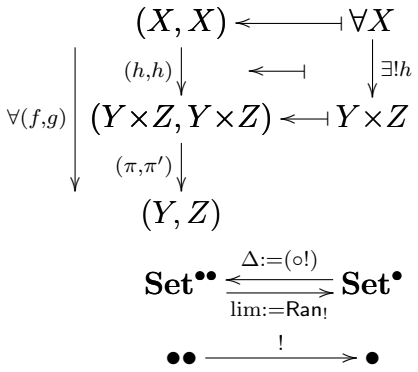
The usual way of defining right Kan extensions is by starting with the functors $F : \mathbf{A} \to \mathbf{B}$ and $H : \mathbf{A} \to \mathbf{C}$ and then saying that a pair (R,T) is *a* right Kan extension of H along F iff it obeys Ran-ness; the functor Ran_F and the adjunction come later. See [Riehl], section 6.1.

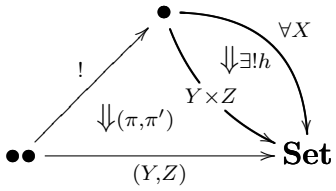
Note that we don't draw the '$\forall V : GF \to H$' in the right half of the diagram — it would overwrite the rest.

8.9 All concepts are Kan extensions

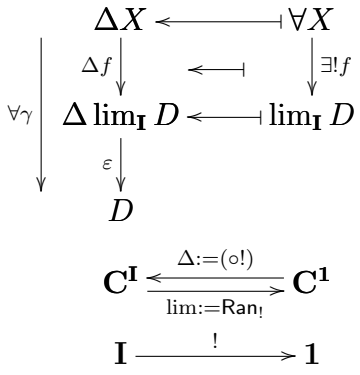
Both [CWM] and [Riehl] have sections called "All concepts are Kan extensions" — section X.7 in [CWM] and 6.5 in [Riehl]. Now that we have a favorite way of drawing right Kan extensions we can use it to draw diagrams for 1) binary products in **Set** are right Kan extensions, 2) limits are right Kan extensions and 3) left adjoints are right Kan extensions. Here they are.

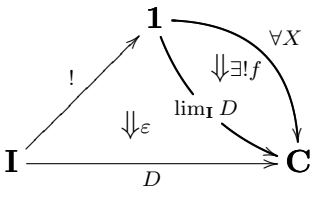
1. Let $\bullet\bullet$ be the discrete category with two objects, $\bullet$ be the discrete category with one object, and $! : \bullet\bullet \to \bullet$ be the unique functor from $\bullet\bullet$ to $\bullet$. Then:





2. Let $\mathbf{I}$ be a finite index category — for example, $\mathbf{I} = \begin{pmatrix} & & 1 \\ & & \downarrow \\ 2 & \to & 3 \end{pmatrix}$ — and let $\mathbf{C}$ be a category with finite limits. A functor $D : \mathbf{I} \to \mathbf{C}$ is a diagram of shape $\mathbf{I}$ in $\mathbf{C}$. Let's denote by $\mathbf{1}$ the discrete category with a single object — the name '$\mathbf{1}$' is more standard than '$\bullet$'. Then:

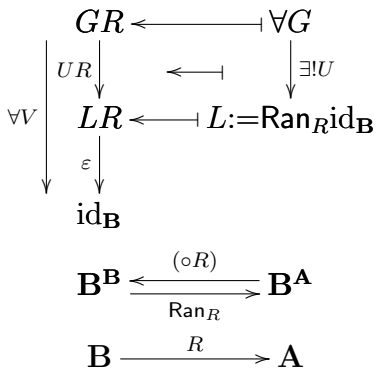


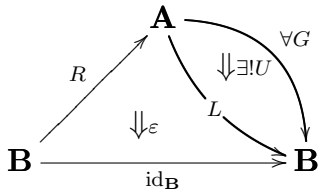


3. Left adjoints are right Kan extensions. If

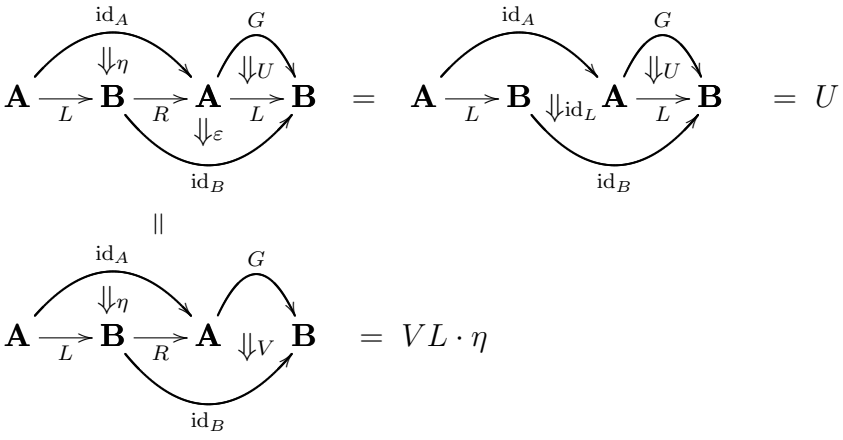
$$\mathbf{B} \underset{R}{\overset{L}{\rightleftarrows}} \mathbf{A}$$

is an adjunction, then (L, ε) is a right Kan extension of $\mathrm{id}_{\mathbf{B}}$ along R. In a more compact notation, $L := \mathsf{Ran}_R \mathrm{id}_{\mathbf{B}}$ — but in this case we only know the action of Ran_R on the object $\mathrm{id}_{\mathbf{B}}$, and we don't know if this Ran_R can be extended to a "real" functor whose domain is the whole of $\mathbf{B}^{\mathbf{B}}$. The diagram is:





To show that this works we have to prove that $\forall V.\exists!U.(\varepsilon \cdot UR = V)$. We will do that by "inverting the equation $\varepsilon \cdot UR = V$":



The solution in $U := VL \cdot G\eta$.

8.10 A formula for Kan extensions

The sections X.3 of [CWM] and 6.2 of [Riehl] discuss a formula for calculating Kan extensions, that defines $\mathsf{Ran}_F H$ as the functor whose action on objects is:

$$B \mapsto \mathrm{Lim}(B{\downarrow}F \xrightarrow{\pi} \mathbf{A} \xrightarrow{H} \mathbf{Set}),$$

and its action on morphisms is "obvious" in the sense of Section 3. I found this formula totally impossible to understand until I finally found a way to visualize what it "meant".

For each object $B \in \mathbf{B}$ the functor $B{\downarrow}F \xrightarrow{H\circ\pi} \mathbf{Set}$ can be regarded as a diagram in **Set** whose shape is the shape of the comma category $B{\downarrow}F$. If $\mathbf{A}$ and $\mathbf{B}$ are finite preorder categories and F is an inclusion then $B{\downarrow}F$ can "inherit its shape" from $\mathbf{A}$; inclusions of preorders are "toy examples" "for children", but they give us some intuition to start with, and they can help us understand the formal version that can handle more general cases.

These are the diagrams for Ran_F as a right adjoint — note that we use **Set** instead of $\mathbf{C}$ to make things less abstract,

$$\begin{array}{ccc} GF & \dashv & G \\ {\scriptstyle V}\downarrow & \leftrightarrow & \downarrow{\scriptstyle U} \\ H & \longmapsto & \mathsf{Ran}_F H \end{array} \quad = R$$

$$\mathbf{Set}^{\mathbf{A}} \underset{\mathsf{Ran}_F}{\overset{\circ F}{\rightleftarrows}} \mathbf{Set}^{\mathbf{B}}$$

$$\mathbf{A} \xrightarrow{F} \mathbf{B}$$

$$\begin{array}{l} RB = \\ (\mathsf{Ran}_F H)B = \\ \mathrm{Lim}(B{\downarrow}F \xrightarrow{\pi} \mathbf{A} \xrightarrow{H} \mathbf{Set}) \end{array}$$

and here are some diagrams to help us understand the comma category $B{\downarrow}F$ — in the compact notation its objects have names like (A, β), but in the more visual notation they are L-shaped diagrams:

$$\begin{array}{ccc} B & & \begin{pmatrix} B \\ \downarrow{\scriptstyle\beta} \\ A \longmapsto FA \end{pmatrix} \\ {\scriptstyle\beta}\downarrow & & {\scriptstyle\alpha}\downarrow \\ A \longmapsto FA & {\scriptstyle\beta'=F\alpha\circ\beta} & \begin{pmatrix} B \\ \downarrow{\scriptstyle\beta'} \\ A' \longmapsto FA' \end{pmatrix} \\ {\scriptstyle\alpha}\downarrow \quad\longmapsto\quad \downarrow{\scriptstyle F\alpha} & & \\ A' \longmapsto FA' & & \end{array}$$

$$\begin{array}{ccccc} (A,\beta) & \longmapsto & A & \longmapsto & HA \\ {\scriptstyle\alpha}\downarrow & \longmapsto & {\scriptstyle\alpha}\downarrow & \longmapsto & \downarrow{\scriptstyle H\alpha} \\ (A',\beta') & \longmapsto & A' & \longmapsto & HA' \end{array}$$

$$\mathbf{A} \xrightarrow{F} \mathbf{B} \qquad B{\downarrow}F \qquad B{\downarrow}F \xrightarrow{\pi} \mathbf{A} \xrightarrow{H} \mathbf{Set}$$

Let's see an example.

If $\mathbf{A} \xrightarrow{F} \mathbf{B}$ is the inclusion $\begin{pmatrix} & & 2 \\ & & \downarrow \\ 5 & \longrightarrow & 6 \end{pmatrix} \to \begin{pmatrix} 1' & \to & 2' \\ \downarrow & & \downarrow \\ 3' & \to & 4' \\ \downarrow & & \downarrow \\ 5' & \to & 6' \end{pmatrix}$,

then $1'{\downarrow}F = \begin{pmatrix} & & \binom{1'}{2\ F2} \\ & & \downarrow \\ \binom{1'}{5\ F5} & \longrightarrow & \binom{1'}{6\ F6} \end{pmatrix}$ and $3'{\downarrow}F = \begin{pmatrix} & & \\ & & \\ \binom{3'}{5\ F5} & \longrightarrow & \binom{3'}{6\ F6} \end{pmatrix}$,

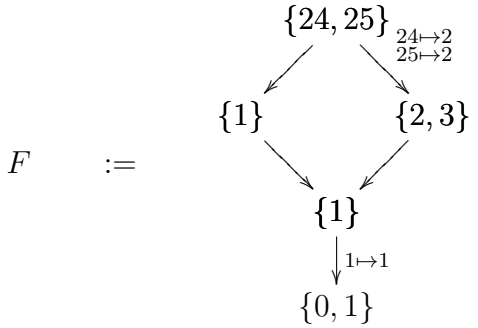
and $(1'{\downarrow}F \xrightarrow{H\circ\pi} \mathbf{Set}) = \begin{pmatrix} & & H_2 \\ & & \downarrow \\ H_5 & \longrightarrow & H_6 \end{pmatrix}$ and $(3'{\downarrow}F \xrightarrow{H\circ\pi} \mathbf{Set}) = \begin{pmatrix} & & \\ & & \\ H_5 & \longrightarrow & H_6 \end{pmatrix}$;

so $R(1') = \mathrm{Lim}(1'{\downarrow}F \xrightarrow{H\circ\pi} \mathbf{Set}) = H_2 \times_{H_6} H_5$,
and $R(3') = \mathrm{Lim}(3'{\downarrow}F \xrightarrow{H\circ\pi} \mathbf{Set}) = H_5$.
We can follow the same pattern to calculate $R(2')$, $R(4')$, $R(5')$, $R(6')$.
The square of the adjunction becomes this, in this particular case:

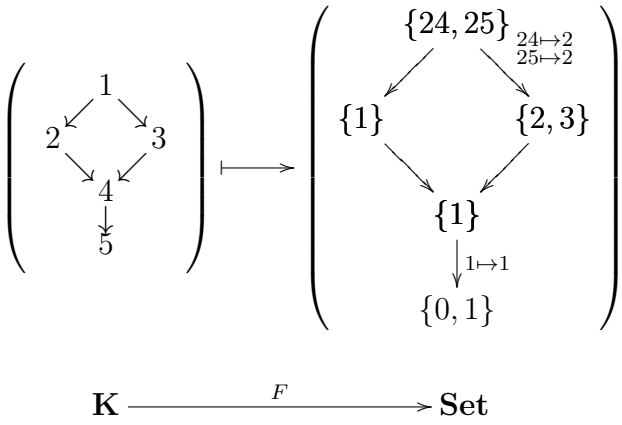
$$\begin{array}{ccc} GF & \dashv & G \\ \downarrow & \leftrightarrow & \downarrow \\ H & \mapsto & \mathsf{Ran}_F H \ = R \end{array} \qquad \begin{array}{ccc} \begin{pmatrix} & & G_2 \\ & & \downarrow \\ G_5 & \longrightarrow & G_6 \end{pmatrix} & \dashv & \begin{pmatrix} G_1 & \longrightarrow & G_2 \\ \downarrow & & \downarrow \\ G_3 & \longrightarrow & G_4 \\ \downarrow & & \downarrow \\ G_5 & \longrightarrow & G_6 \end{pmatrix} \\ \downarrow & \leftrightarrow & \downarrow \\ \begin{pmatrix} & & H_2 \\ & & \downarrow \\ H_5 & \longrightarrow & H_6 \end{pmatrix} & \mapsto & \begin{pmatrix} H_2 \times_{H_6} H_5 & \longrightarrow & H_2 \\ \downarrow & & \downarrow \\ H_5 & \longrightarrow & H_6 \\ \downarrow & & \downarrow \\ H_5 & \longrightarrow & H_6 \end{pmatrix} \end{array}$$

8.11 Functors as objects

One way to treat a diagram in **Set** like this



as a functor is to think that that diagram is an abbreviation — it is just the upper-right part of a diagram like this,



where we add the extra hint that the index category **K** is exactly the kite-shaped preorder category drawn above the "**K**".

The convention (CFSh) says that the image by a functor of a diagram is a diagram with the same shape, so according to that convention we have $F(1) = \{24, 25\}$, $F(4 \to 5) = (\{1\} \overset{1\mapsto 1}{\to} \{0, 1\})$, and so on; so the upper right part of the diagram above *defines* F.

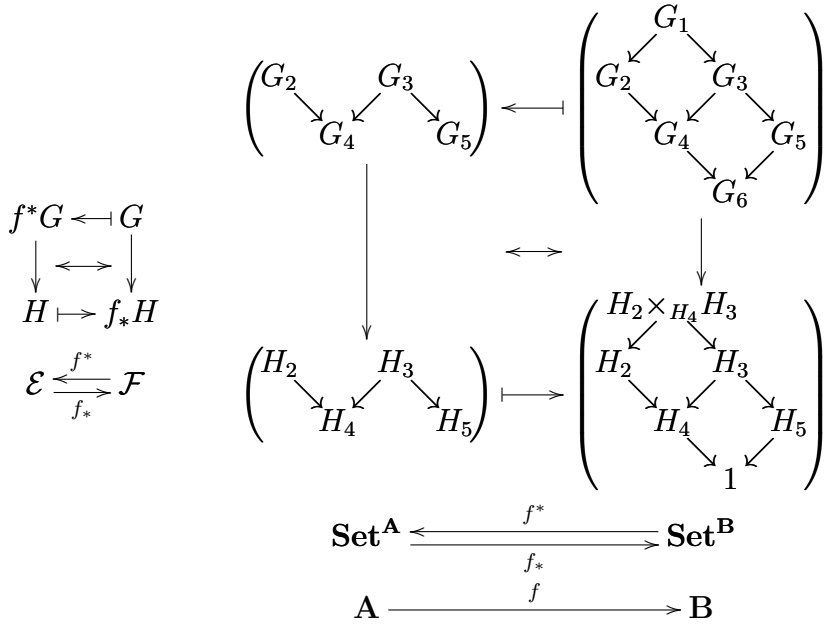
Note that the single '$\mapsto$' above the $\mathbf{K} \overset{F}{\to} \mathbf{Set}$ stands for several '$\mapsto$'s, one for each object and one for each morphism, and note that F is an object of $\mathbf{Set}^{\mathbf{K}}$.

8.12 Geometric morphisms for children

Let $\mathbf{A}$ and $\mathbf{B}$ be these preorder categories, and let $f : \mathbf{A} \to \mathbf{B}$ be the inclusion functor from $\mathbf{A}$ to $\mathbf{B}$:

$$A := \begin{pmatrix} 2 & & 3 & & \\ & \searrow & \swarrow & \searrow & \\ & & 4 & & 5 \end{pmatrix} \qquad B := \begin{pmatrix} & & 1 & & \\ & \swarrow & & \searrow & \\ 2 & & & & 3 \\ & \searrow & & \swarrow & \searrow \\ & & 4 & & 5 \\ & & \searrow & \swarrow & \\ & & & 6 & \end{pmatrix}$$

The left half of the diagram below is the standard definition of a geometric morphism f from a topos $\mathcal{E}$ to a topos $\mathcal{F}$. A geometric morphism $f : \mathcal{E} \to \mathcal{F}$ is actually an adjunction $f^* \dashv f_*$ plus the guarantee that $f^* : \mathcal{E} \leftarrow \mathcal{F}$ preserves limits, which is a condition slightly weaker than requiring that f^* has a left adjoint. When that left adjoint exists it is denoted by $f^!$, and we say that $f^! \dashv f^* \dashv f_*$ is an *essential geometric morphism*. The only non-standard thing about the diagram at the left below is that is contains an internal view of the adjunction $f^* \dashv f_*$.



The right half of the diagram is a particular case of the left half. Its lower line, $\mathbf{A} \xrightarrow{f} \mathbf{B}$, does not exist in the left half. The inclusion functor f induces

adjunctions $f^! \dashv f^* \dashv f_*$ as this,

$$\mathbf{Set}^{\mathbf{A}} \underset{f_*}{\overset{f^!}{\underset{\longleftarrow}{\overset{\longrightarrow}{\longrightarrow}}}} \mathbf{Set}^{\mathbf{B}}$$

$$\mathbf{A} \xrightarrow{\quad f \quad} \mathbf{B}$$

where f^* is easy to define and $f^!$ and f_* not so much — the standard way to define $f^!$ and f_* is by Kan extensions.

The big square in the upper part of the diagram is an internal view of the adjunction $f^* \dashv f_*$, with the functors f^*G, G, H, and f_*H being displayed as their internal views. We can choose the sets $G_1, \ldots, G_6$ and the morphisms between them arbitrarily, so this is an internal view of an arbitrary functor $G : \mathbf{B} \to \mathbf{Set}$; and the same for H.

The arrow $f^*G \leftarrow\!\shortmid G$ can be read as a definition for the action of f^* on objects — it just erases some parts of the diagram — and the arrow $H \mapsto f_*H$ can be read as a definition for the action of f_* on objects — f_* "reconstructs" H_1 and H_6 in a certain natural way. It is easy to reconstruct the actions of f^* and f_* on morphisms from just what is shown, and to reconstruct the two directions of the bijection.

The big diagram above can be used 1) to convince categorists who are not seasoned toposophers that this diagrammatic language can make some difficult categorical concepts more accessible, and 2) as a starting point to generate diagrams "for children" for several parts of the Elephant ([Elephant]), and even to prove new theorems on toposes. For more on (1), see [OL18] and [Och18]; for (2), see [MDE].

9 Related and unrelated work

The diagrammatic language that I described here seems to be unrelated to the ones in [CK17] and [Coe11] — that describe *lots* of diagrammatic languages — and also unrelated to [Mar14]. We lower the level of abstraction — see for example Section 8.7 — while they (usually) raise it.

I've taken an approach that is the opposite of [CW01] and [Các04]. Cáccamo and Winskel define a derivation system that can only construct functors, natural transformations, etc, that obey the expected naturality conditions, while we allow some kinds of sloppinesses, like constructing something

that looks like a functor and pretending that it is a functor when it may not be. When I started working on this diagrammatic language I had a companion derivation system for it; [IDARCT, Section 14] mentions it briefly, but it doesn't show the introduction rules that create (proto)functors and (proto)natural transformations and that allow being sloppy ("in the syntactical world"). That derivation system was incomplete in all senses — it even had "rules" that I knew how to apply in particular cases but I didn't know how to formalize.

Some of my excuses for allowing one to pretend that a functor is a functor and leaving the verification to a second stage come from [Che04]. I learned a *lot* on how mathematicians use intuition and diagrams from [Krö07] — [Krö18] is a great summary — and [Cor04], and they have helped me to identify which characteristics of my diagrammatic language are very unusual and may be new, and that deserve to be presented in detail.

Many of the first ideas for my diagrammatic language appeared when I was reading [See83], [See84], [See87], [Jac99], and [BCS06] and trying to draw the "missing diagrams" in those papers in both the original notation and in the "archetypal case" ([IDARCT, Section 16]).

Many of the later ideas appeared when I was trying to understand sheaves using a certain approach "for children" ([PH2]): I learned how to draw diagrams showing a Grothendieck topology, its corresponding Lawvere-Tierney topology, and its corresponding nucleus *in particular cases*, and I knew that *there had to be a way* (in the sense of [Che04]) to "lift" these diagrams of the correspondences in particular cases to a diagram of the correspondences in the general case... the details of this "lifting" were hard to formalize, but missing details started to become clear when I required the diagrams to be translatable to a pseudocode that could be translated to Agda. At this moment [PH2] is still incomplete, but some of the ideas in DL were motivated by its conceptual holes.

References

[AgdaMan] The Agda Team. *Agda User Manual*.

[AT11] S. Abramsky and N. Tzevelekos. "Introduction to Categories and Categorical Logic". https://arxiv.org/pdf/1102.1313.pdf. 2011.

[Awo06] S. Awodey. *Category Theory*. Oxford, 2006.

[Bad09] A. Badiou. *Logics of Worlds - Being and Event, 2*. Continuum, 2009.

[Bad14] A. Badiou. *Mathematics of The Transcendental*. Bloomsbury, 2014.

[BCS06] R. Blute, R. Cockett, and R. A. G. Seely. "Differential Categories". In: *Mathematical Structures in Computer Science* 16 (2006). http://www.math.mcgill.ca/rags/difftl/difftl.pdf, pp. 1049–1083.

[Các04] M. J. Cáccamo. "A Formal Calculus for Categories". http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.1.7460. PhD thesis. Aarhus, 2004.

[Car03] A. Carson. *If Not, Winter – Fragments of Sappho*. Vintage Books, 2003.

[Che04] E. Cheng. "Mathematics, Morally". http://eugeniacheng.com/wp-content/uploads/2017/02/cheng-morality.pdf. 2004.

[CK17] B. Coecke and A. Kissinger. *Picturing Quantum Processes. A First Course in Quantum Theory and Diagrammatic Reasoning*. http://www.cambridge.org/gb/pqp. Cambridge, 2017.

[Coe11] B. Coecke, ed. *New Structures for Physics*. https://www.springer.com/br/book/9783642128202. Springer, 2011.

[Cor04] D. Corfield. *Towards a Philosophy of Real Mathematics*. Cambridge, 2004.

[CW01] M. J. Cáccamo and G. Winskel. "A Higher-Order Calculus for Categories". https://www.brics.dk/RS/01/27/BRICS-RS-01-27.pdf. 2001.

[CWM] S. Mac Lane. *Categories for the Working Mathematician (2nd ed.)* Springer, 1997.

[Elephant] P. T. Johnstone. *Sketches of an Elephant: A Topos Theory Compendium*. Vol. 1. Oxford, 2002.

[ES52] S. Eilenberg and N. Steenrod. *Foundations of algebraic topology*. Princeton, 1952.

[Freyd76] P. Freyd. "Properties Invariant within Equivalence Types of Categories". In: *Algebra, Topology and Category Theory: A Collection of Papers in Honour of Samuel Eilenberg*. Ed. by A. Heller and M. Tierney. http://angg.twu.net/Freyd76.html. Academic Press, 1976, pp. 55–61.

[FS19] B. Fong and D. I. Spivak. *Seven Sketches in Compositionality: An Invitation to Applied Category Theory*. https://arxiv.org/pdf/1803.05316.pdf. Cambridge, 2019.

[FS90] P. Freyd and A. Scedrov. *Categories, Allegories*. North-Holland, 1990.

[Gan13] M. Ganesalingam. *The Language of Mathematics - A Linguistic and Philosophical Investigation*. Springer, 2013.

[Haz19] S. Hazratpour. "A Logical Study of Some 2-Categorical Aspects of Topos Theory". PhD thesis. Birmingham, 2019.

[HC20] J. Hu and J. Carette. "Formalizing Category Theory in Agda". https://arxiv.org/abs/2005.07059. 2020.

[IDARCT] E. Ochs. "Internal Diagrams and Archetypal Reasoning in Category Theory". In: *Logica Universalis* 7.3 (Sept. 2013). http://angg.twu.net/math-b.html#idarct, pp. 291–321.

[Jac99] B. Jacobs. *Categorical Logic and Type Theory*. Studies in Logic and the Foundations of Mathematics 141. North-Holland, Elsevier, 1999.

[Jam01] M. Jamnik. *Mathematical Reasoning with Diagrams: From Intuition to Automation*. CSLI, 2001.

[JIB22] P. Jansson, C. Ionescu, and J.-P. Bernardy. *Domain-Specific Languages of Mathematics*. https://github.com/DSLsofMath/DSLsofMath. College Publications, 2022.

[Krö07] R. Krömer. *Tool and Object: A History and Philosophy of Category Theory*. https://www.springer.com/gp/book/9783764375232. Birkhäuser, 2007.

[Krö18] R. Krömer. "Category theory and its foundations: the role of diagrams and other "intuitive" material". Slides for his keynote talk at the UniLog 2018. http://angg.twu.net/logic-for-children-2018/ralf_kroemer__slides.pdf. 2018.

[Leinster] T. Leinster. *Basic Category Theory*. https://arxiv.org/pdf/1612.09375.pdf. Cambridge, 2014.

[LR03] W. Lawvere and R. Rosebrugh. *Sets for Mathematics*. Cambridge, 2003.

[MacLaneNotes] S. Mac Lane. "Lectures on Category Theory (Bowdoin Summer School)". Handwritten notes by E. Cooper: https://ncatlab.org/nlab/show/Lectures+on+category+theory. 1969.

[Mar14] D. Marsden. "Category Theory Using String Diagrams". https://arxiv.org/pdf/1401.7220.pdf. 2014.

[MDE] E. Ochs. "On Some Missing Diagrams in The Elephant". http://angg.twu.net/math-b.html#missing-diagrams-elephant. 2019.

[Mil20] B. Milewski. "Category Theory for Programmers, OCaml Edition". https://github.com/hmemcpy/milewski-ctfp-pdf/releases/download/v1.4.0-rc1/category-theory-for-programmers-ocaml.pdf. 2020.

[NC08] U. Norell and J. Chapman. "Dependently Typed Programming in Agda". http://www.cse.chalmers.se/~ulfn/papers/afp08/tutorial.pdf. 2008.

[NG14] R. Nederpelt and H. Geuvers. *Type Theory and Formal Proof - An Introduction*. Cambridge, 2014.

[Och18] E. Ochs. "Visualizing Geometric Morphisms". http://angg.twu.net/LATEX/2018vichy-vgms-slides.pdf. 2018.

[Och19] E. Ochs. "How to almost teach Intuitionistic Logic to Discrete Mathematics Students". http://angg.twu.net/LATEX/2019logicday.pdf. 2019.

[Och20] E. Ochs. "What kinds of knowledge do we gain by doing Category Theory in several levels of abstraction in parallel?" http://angg.twu.net/math-b.html#2020-tallinn. 2020.

[Och22] E. Ochs. "On the missing diagrams in Category Theory: Agda code". Work in progress! See http://angg.twu.net/math-b.html#md. 2022.

[OL18] E. Ochs and F. Lucatelli. "Logic for Children - Workshop at UniLog 2018 (Vichy) - unofficial homepage". http://angg.twu.net/logic-for-children-2018.html. 2018.

[Penrose] K. Ye et al. "Penrose: From Mathematical Notation to Beautiful Diagrams". In: http://penrose.ink/media/Penrose_SIGGRAPH2020.pdf. 2020.

[Perrone] P. Perrone. "Notes on Category Theory with examples from basic mathematics". https://arxiv.org/abs/1912.10642. 2020.

[PH1] E. Ochs. "Planar Heyting Algebras for Children". In: *South American Journal of Logic* 5.1 (2019). http://angg.twu.net/math-b.html#zhas-for-children-2, pp. 125–164.

[PH2] E. Ochs. "Planar Heyting Algebras for Children 2: Local J-Operators, Slashings, and Nuclei". http://angg.twu.net/math-b.html#zhas-for-children-2. 2021.

[Pow90] A. J. Power. "A 2-Categorical Pasting Theorem". In: *Journal of Algebra* 129.2 (1990). https://core.ac.uk/download/pdf/81929927.pdf, pp. 439–445.

[Riehl] E. Riehl. *Category Theory in Context*. http://www.math.jhu.edu/~eriehl/context.pdf. Dover, 2016.

[See83] R. A. G. Seely. "Hyperdoctrines, natural deduction, and the Beck condition". In: *Zeitschrift f. math. Logik und Grundlagen d. Math.* 29 (1983). http://www.math.mcgill.ca/rags/ZML/ZML.PDF, pp. 505–542.

[See84] R. A. G. Seely. "Locally Cartesian closed categories and type theory". In: *Math. Proc. Cambridge Philos. Soc.* 95.1 (1984). http://www.math.mcgill.ca/rags/LCCC/LCCC.pdf, pp. 33–48.

[See87] R. A. G. Seely. "Categorical Semantics for Higher Order Polymorphic Lambda Calculus". In: *Journal of Symbolic Logic* 52.4 (1987). http://www.math.mcgill.ca/rags/JSL/PLC.pdf, pp. 969–988.

[Sel13] P. Selinger. "Lecture Notes on the Lambda Calculus". https://arxiv.org/abs/0804.3434v2. 2013.

[SICP] H. Abelson and G.J. Sussman. *Structure and Interpretation of Computer Programs, 2nd ed.* https://web.mit.edu/alexmv/6.037/sicp.pdf. MIT, 1996.

[Tay99] P. Taylor. *Practical Foundations of Mathematics*. Cambridge, 1999.

[TG88] S. P. Thompson and M. Gardner. *Calculus Made Easy: being a very-simplest introduction to those beautiful methods of reckoning which are generally called by the terrifying names of the differential calculus and the integral calculus*. St. Martin's Press, 1988.

[WKS20] P. Wadler, W. Kokke, and J. G. Siek. *Programming Language Foundations in Agda*. July 2020. URL: http://plfa.inf.ed.ac.uk/20.07/.