Internal Diagrams and Archetypal Reasoning in Category Theory

Eduardo Ochs

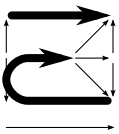
Abstract. We can regard operations that discard information, like specializing to a particular case or dropping the intermediate steps of a proof, as *projections*, and operations that reconstruct information as *liftings*. By working with several projections in parallel we can make sense of statements like "**Set** is the archetypal Cartesian Closed Category", which means that proofs about CCCs can be done in the "archetypal language" and then lifted to proofs in the general setting. The method works even when our archetypal language is diagrammatical, has potential ambiguities, is not completely formalized, and does not have semantics for all terms. We illustrate the method with an example from hyperdoctrines and another from synthetic differential geometry.

Mathematics Subject Classification (2010). Primary 18C50; Secondary 03G30.

1. Mental Space and Diagrams

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 $\mathsf{Frob}^\natural : \Sigma_f(P \wedge f^*Q) \xrightarrow{\cong} \Sigma_f P \wedge Q$ 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.

I cannot yet define precisely what it is to "think diagramatically", but for the purposes of this paper a loose definition — or rather, a set of concepts and techniques for diagrammatical thinking, plus non-trivial consequences of "thinking" in that way — shall do. I will resort to a narrative device: *I will speak as from a semi-fictional "I" who thinks as diagramatically as possible, and who always uses diagrams as a help for his bad algebraic thinking.*

2. Projections and Liftings

Take the concept of "projection", from the realm of covering spaces or from Linear Algebra. A projection map discards information — coordinates, maybe — and may collapse objects that were originally distinct.

I try to organize my diagrams to make the projection arrows — most of them, at least — go down. Figure 1 is an example.

Specialization acts like a projection. Look at the top arrow in Figure 1: except for the choice of a particular value for n, we have only lost information. *Discarding intermediate steps*, as in the bottom arrow, is a kind of *erasing*, and erasings are evidently projections.

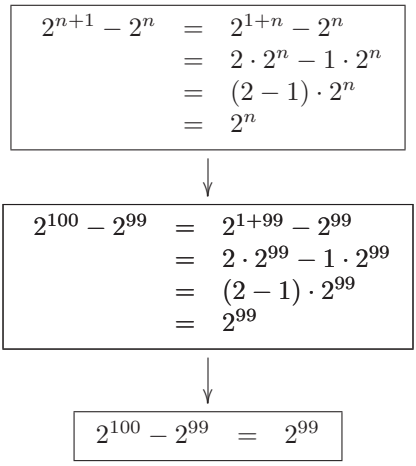


FIGURE 1. A specialization and an erasing.

The opposite of *to project* is *to lift*; projecting is easy, lifting is hard. A projection map, $p : E \to B$, may have any number of preimages for a point b in the base space — one, many, none —, and, to make things worse, we will often be interested in very specific cases where we are somehow able to recognize whether a given lifting is "good" or not — for example, we may be looking for the "right" generalization for $2^{100} - 2^{99} = 2^{99}$ —, but where we are unable to define exactly what a "good lifting" is.

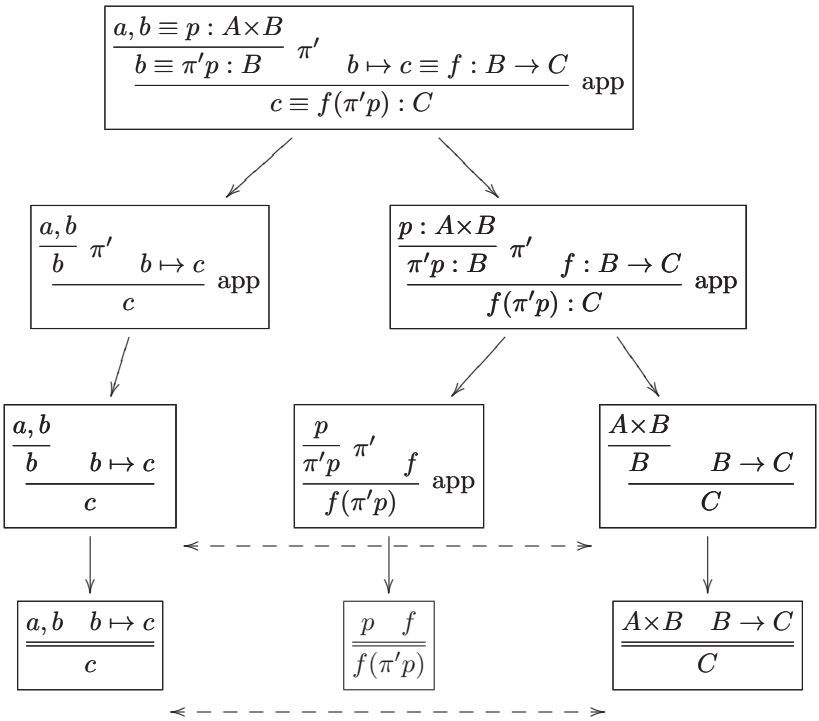


FIGURE 2. Downcased types.

3. Downcased Types

Suppose that we have a point $p \in A \times B$ and a function $f : B \to C$. There is a single "obvious" way to build a point of C starting from this data. It is "obvious" because we have a search method that finds it; it is related to proof search (*via* the Curry-Howard isomorphism). Here's how it works, briefly and informally.

A point $p \in A \times B$ is a pair made of an 'a' and a 'b'. If we allow "long names" for variables we can replace the 'p' by another name, that reflects its type more closely; so let's rename 'p' to 'a, b'. Similarly, an $f : B \to C$ is an operation that takes each 'b' to a 'c', so let's rename 'f' to '$b \mapsto c$'. Now, with this new notation, we are looking for an operation that takes an 'a, b' and a '$b \mapsto c$' and produces a term that "deserves the name" 'c'. We start at the tree in the lower right rectangle in Figure 2, and we want to arrive at the tree in the lower left; both trees have the same shape, and by relating them we will see that the term corresponding to c is $f(\pi' p)$. Note that in this new language — "Downcased Types" — we have no syntactical distinction between variables and non-atomic terms.

All the solid arrows in Figure 2 are erasings. The dashed arrows are "uppercasings" when we go rightwards, "downcasings" when we go to the left; note that '$\times$' is downcased to ',', and '$\to$' is downcased to '$\mapsto$'. If we start at the lower left and we move right through the dashed arrow, we get the types of the (three) objects involved; what we still need to do from there is a kind of "term inference"...

"Type inference" is very well-known, and polymorphic languages like Haskell and ML implement algorithms for it; "term inference", on the other hand, is rarely mentioned in the literature, but techniques like parametricity ([Wad89], [BJP10]) provide useful meta-theorems about properties that all possible inferrable terms must obey. It would be lovely if these techniques could do term inference for us, algorithmically — but they can't, so when we need to infer terms we usually do the work by hand.

As term inference is hard, let's turn our attention to something easier, and looser. "Inference" carries the connotation of something that can be done algorithmically, without any previous knowledge of the result. We will focus on the process of "reconstructing" the desired term, $c \equiv f(\pi' p)$, from the downcased tree in the lower left of Figure 2. A "reconstruction" may need hints to be completed, and may depend on making the right choices at some points, motivated by unformalizable bits of common sense, or by intuition, hindsight, or by vague rememberances, by a sense of good style, or whatever else. A "reconstruction" is the result of an *incomplete* algorithm (or a "method", rather than an "algorithm"); when we perfect a method for reconstruction, and finish filling up all its gaps, it becomes an "inference algorithm".

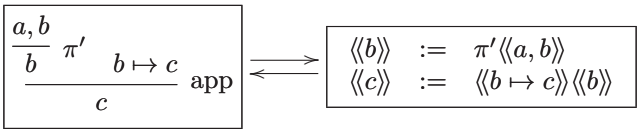
For our purposes, "reconstruction" will be enough — and real "inference" will be close to impossible.

4. The Dictionary

The downcased notation has to be used with care, as it doesn't come with any built-in devices to protect us from ambiguities. For example, we could have tried to find a term "deserving the name" '$a, a \mapsto a, a$' — and there are four different ones!...

One way to avoid this problem is to consider that the downcased "names" are just that, *names*, and that we have a *dictionary* that associates to each name its *meaning*.

In the case of λ-calculus, a dictionary relating each downcased name to its meaning can be extracted from a derivation tree (if all the rule names are present), and the derivation tree can be reconstructed from its associated dictionary. For example,



and if we add to that dictionary the entries

$$\begin{array}{rcl} \langle\!\langle a, b \rangle\!\rangle & := & p \\ \langle\!\langle b \mapsto c \rangle\!\rangle & := & f \end{array}$$

then we can reconstruct the tree

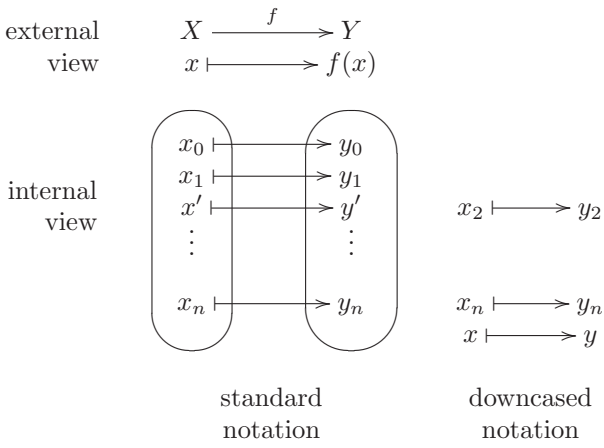
$$\dfrac{\dfrac{p}{\pi' p}\ \pi' \qquad f}{f(\pi' p)}\ \text{app}$$

from that. Note that we use double angle brackets, $\langle\!\langle \cdot \rangle\!\rangle$, to separate names from one another and to distinguish them from standard notation, and that we use '$\equiv$' to indicate change of notation — usually between downcased and standard.

5. Internal Diagrams

Several of the initial ideas for the system of downcased types came from attempts to formalize something that I will call "physicist's notation", and that should be familiar to most readers; I have never seen any detailed formalizations of it, though.

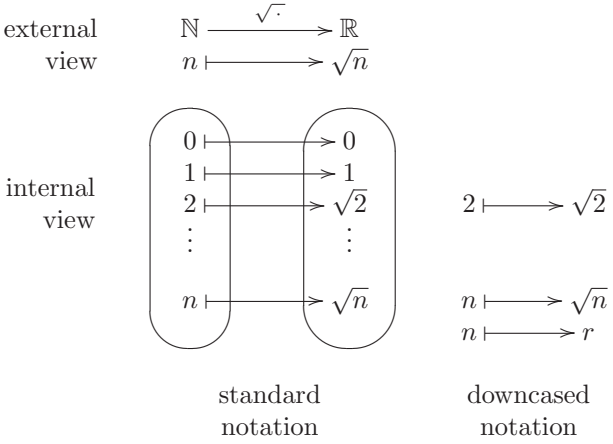
Suppose that we have a function $f : \mathbb{R} \to \mathbb{R}$, and we draw the graph of $y = f(x)$. Then, given points x_0, x_1, x_2, x' on the "x-axis" we have default meanings for the names y_0, y_1, y_2, y': namely, $y_0 := f(x_0)$, $y' := f(x')$, and so on. It makes sense to write X for the domain of f and Y for its codomain, and if we draw the "internal diagram" (as in [LS97], p.14) of the map f, we get this:



Note that for each name like x_2 of a point in the x-axis (the "space of 'x's", X) we have chosen a similar name for a point in the y-axis (the "space of 'y's", Y). We will call the implicit operation on names the *syntactical action* of the function $f \equiv x \mapsto y$; in the case above, the syntactical action replaced the 'x' of each original name into a 'y', and kept all the "decorations" unchanged.

The syntactical action does not need to be defined for all possible names of points in X — in fact, we restrict ourselves to "good" names of points in X exactly to make the syntactical action easier to describe.

Now let's take a function less abstract: $\sqrt{\cdot} : \mathbb{N} \to \mathbb{R}$. Its action is implicit in its name ("$\sqrt{\cdot}$"), and if we examine its internal diagram,



we may conclude that it should be possible to use '$n \mapsto \sqrt{n}$' as its name; or even '$2 \mapsto \sqrt{2}$' — "the function that takes 2 to $\sqrt{2}$, for every value of '2'"...

As the reader may have guessed, there is no clear separation between what are "good names" and "bad names" for an object; instead we have a murky line. We have just seen in the first example that 'x_0', 'x'', etc, can all be downcasings for 'X', and maybe the '1' in second example, if taken as a name, should be uppercaseable both as $\mathbb{N}$ and as $\mathbb{R}$... The trick is all in our use of the dictionary: we *can* define the meaning of '$n \mapsto \sqrt{n}$' to be $\sqrt{\cdot} : \mathbb{N} \to \mathbb{R}$, and, if we feel that $\#^!_\$$ is a good name for the square root, we can define it to stand for the square root too...

6. Parallel Notations

We are using two notations — downcased and standard — in parallel; it is possible to use more. One way to visualize what is going on is to think in terms of computer interfaces. Imagine a user inspecting huge data structures by displaying them on a computer screen in several forms — he can toggle switches that control what gets shown and what is omitted.

The downcased notation of the previous sections is *my* compromise between *my* intuition and a formalization. Suppose that someone has created a third notation, that he claims that is much closer to *his* intuitions. He can set the controls to display only his favorite notation, as in the left side of the diagram below; or he can display all together at the same time — as at the top.

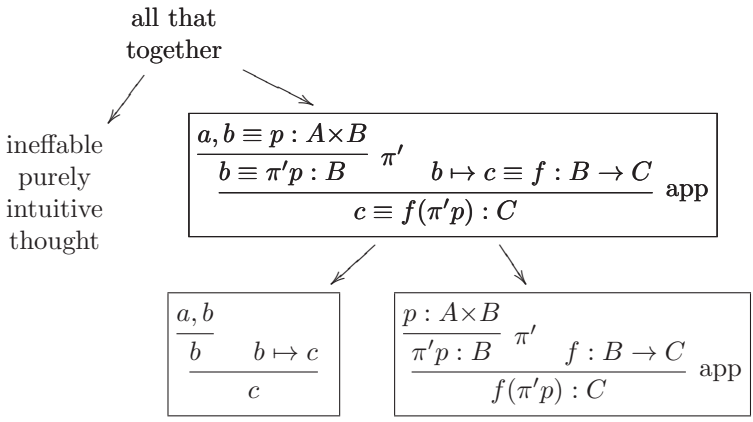


FIGURE 3. Parallel notations.

Working with three parallel notations is similar to working with two.

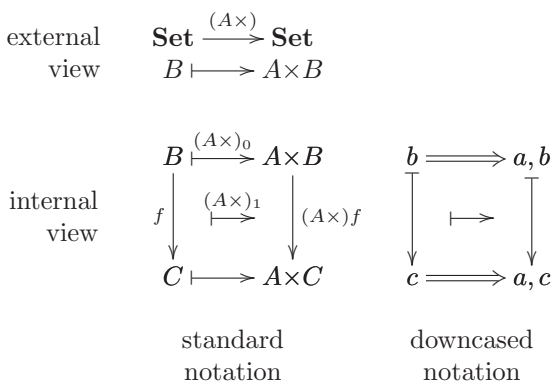
At this point our two notations, downcased and standard, generate *trees* with exactly the same shapes; in the next section we will begin to compare *diagrams* done in different notations, but having the same shapes; and starting on section 13 the parallelism will be looser — there will be correspondences between trees and dictionaries, on one side, and "strictly 2-dimensional" categorical diagrams, on another. In all cases it is convenient to work in a "projected view", yet pretend that we are in the situation with total information — that the rest of the information "is there", but hidden.

7. Functors

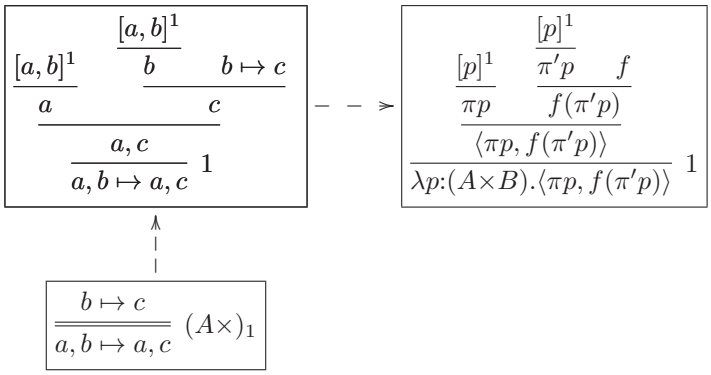
Fix a set A. It induces a functor $(A\times)$, that takes each set B to $A \times B$ and each function $f : B \to C$ to a function $(A\times)f : A \times B \to A \times C$. Let's use the subscripts '0' and '1' to distinguish the two actions of a functor: $(A\times)_0$ is its action on objects (sets, in this case), $(A\times)_1$ is its action on morphisms (i.e., on functions).

It is quite common in the literature of Category Theory to see things like: "let $(A\times)$: **Set** $\to$ **Set** the functor that takes each set B to $A \times B$". The action on morphisms is not described — it is "obvious", and it is left to the reader to discover. The reader should apply a kind of search algorithm to find it. Which algorithm is this?

Let's downcase this problem. The action on objects takes each "space of 'b's" to a "space of 'a, b's"; its syntactical action is to prepend an 'a,'. The action on morphisms takes each $b \mapsto c$ to an $a, b \mapsto a, c$ — i.e., the syntactical action on morphisms consists of applying the syntactical action on objects to



(A) Downcasing the functor $(A\times)$.



(B) A way to reconstruct the "obvious" part.

FIGURE 4. Functors.

both the domain and the codomain. *This is a general pattern:* in all functors the two syntactical actions will be related in this way.

The diagram in Figure 4a is very similar to the ones in the previous section, but there the blobs were sets and they had points; now the blobs — which we are no longer drawing — are categories, that have objects, and between these objects we may have *morphisms*. Also, the downcasing is changing the notation more than before, and we are downcasing the '$\mapsto$'s that were functor actions as '$\Rightarrow$'. The '$\Rightarrow$' is to remind us that: 1) something non-obvious is going on — points of B are not being taken to points of $A\times B$, instead the whole set B was fed into $(A\times)_0$, and we got back $A\times B$ —, and 2) that '$b \Rightarrow a,b$' is *two* actions.

So, we know just the "name" of the action on morphisms of this functor. How do we reconstruct its "meaning"? The answer is obtained by liftings, as in Figure 4b; we find that $(A\times)f := \lambda p{:}A\times B.\langle \pi p, f(\pi' p)\rangle$.

8. Natural Transformations

Now fix two sets, A and A', and a map $\alpha : A \to A'$. We have natural constructions for functors $(A\times)$ and $(A'\times)$, and, besides that, a natural transformation ("NT", from now on), $(\alpha\times)$, going from $(A\times)$ to $(A'\times)$. An NT has a *single* action, that takes *objects* to *morphisms*.

We will need a convention. If X and Y are objects of a category $\mathbf{C}$, then both $f : X \to Y$ and $X \xrightarrow{f} Y$ will denote a morphism, but $X \to Y$ will denote the full hom-set $\mathrm{Hom}_{\mathbf{C}}(X, Y)$. With that convention, if $F, G : \mathbf{A} \to \mathbf{B}$ are functors and $T : F \dot{\to} G$ is a NT, we can represent the internal view of T as:

$$A \mapsto (FA \xrightarrow{TA} GA).$$

Let's take the convention up one level: $A \dot{\to} (FA \xrightarrow{TA} GA)$ will denote a specific NT, but $A \dot{\to} (FA \to GA)$ will mean the class of all NTs from F to G; and we downcase '$\dot{\to}$' as '$\overset{\bullet}{\to}$'; see Figures 5a and 5b.

We know the "syntactical action" of $(\alpha\times)$: it is $B \mapsto (A\times B \xrightarrow{\alpha\times B} A'\times B)$ in standard notation, $b \overset{\bullet}{\to} (a, b \mapsto a', b)$ after downcasing. To obtain a "meaning" for that we can apply the same procedure as in the previous section; we discover that $\alpha\times B \;:=\; \lambda p : A\times B.\langle f(\pi p), \pi' p\rangle$.

Now let's look at the notations for the general case. Let $F, G : \mathbf{A} \to \mathbf{B}$ be functors, and $T : F \dot{\to} G$ be an NT between them. Everything is similar, but we need a way to downcase the objects FA and GA... a good choice is as 'a^F' and 'a^G' — because the idea of "an a^F" *suggests nothing at all*, and so it reminds us that the functors F and G may be abstract.

We often have to deal with categories whose objects don't have any reasonable notion of "elements". If our $\mathbf{B}$ is a category like that, then $a^F \mapsto a^G$ will be a name for a morphism, but the names 'a^F' and 'a^G' "will not have semantics" — our dictionary will not attribute any meanings to them. See [Krö07], especially its sections 3.3.4.4 and 5.3.2.1, for a discussion; our downcased notation is, in a sense, "a language for diagram chasing".

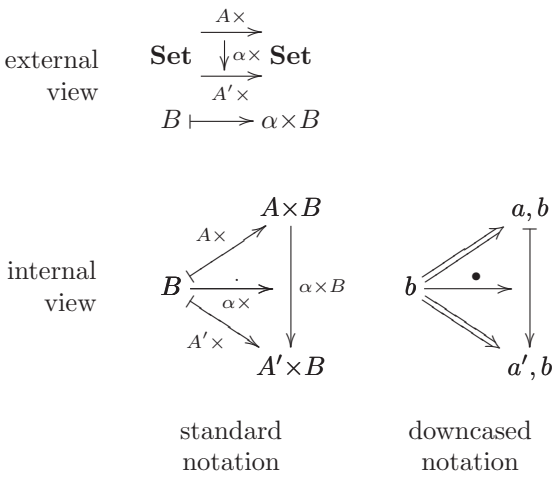
9. Adjunctions

Fix an object B of **Set**, and let's write $B{\to}C$ for C^B. Then we have a functor $(B{\to}) : \mathbf{Set} \to \mathbf{Set}$, whose syntactical action is $C \mapsto (B{\to}C)$, in standard notation, and $c \Rightarrow b \mapsto c$ after downcasing. This functor is right adjoint to $(\times B) : \mathbf{Set} \to \mathbf{Set}$, and we will represent that diagrammatically as a square: Figure 6a.

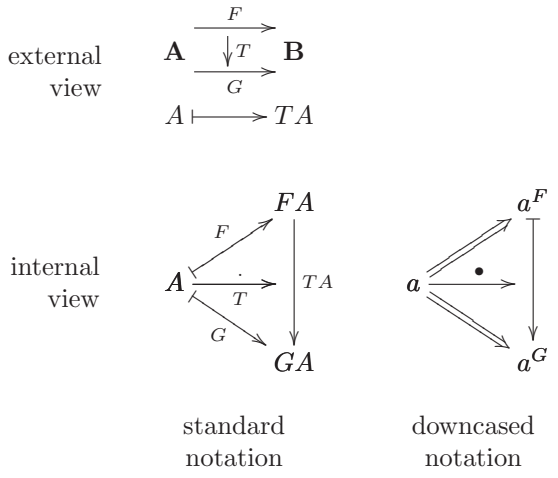
The two transpositions, cur and uncur, are natural transformations with actions:

$$(A^{\mathrm{op}}, C) \mapsto ((A\times B \xrightarrow{f} C) \mapsto (A \xrightarrow{\mathrm{cur}\, f} (B{\to}C)))$$
$$(A^{\mathrm{op}}, C) \mapsto ((A \xrightarrow{g} (B{\to}C)) \mapsto (A\times B \xrightarrow{\mathrm{uncur}\, g} C))$$

where A^{op} is an object of $\mathbf{Set}^{\mathrm{op}}$, and (A^{op}, C) is an object of $\mathbf{Set}^{\mathrm{op}} \times \mathbf{Set}$.



(A) The NT $(\alpha\times) : (A\times) \dot{\to} (A'\times)$.

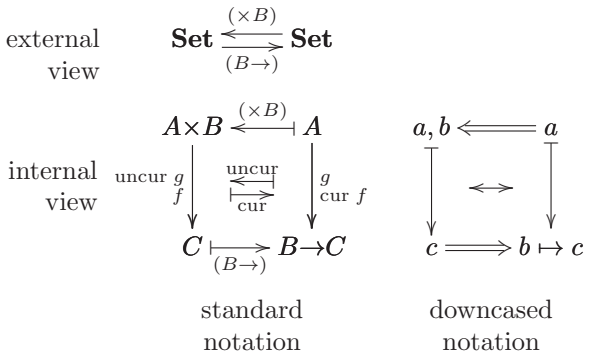


(B) The generic case: $T : F \dot{\to} G$.

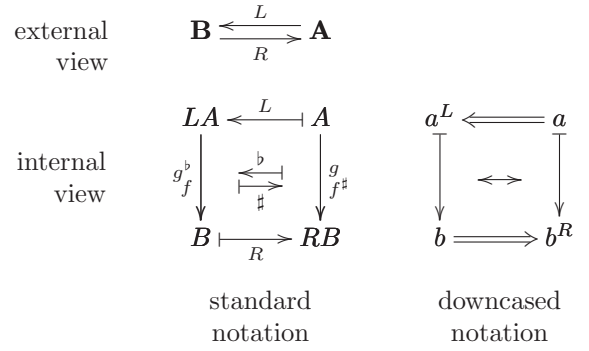
FIGURE 5. Natural transformations

The notation in the general case is similar. An adjunction $L \dashv R$ is represented diagrammatically as the square in Figure 6b.

We will usually draw L as going left, R as going right, and call the transpositions '♭' and '♯'. Note that in a square drawn in this way the adjunction $L \dashv R$ itself, viewed "externally", would be something vertical: $\begin{smallmatrix} L \\ \dashv \\ R \end{smallmatrix}$ — but its maps between hom-sets are horizontal arrows.



(A) The adjunction $(\times B) \dashv (B\to)$.



(B) Notation for a general case: $L \dashv R$.

FIGURE 6. Adjunctions.

Just as a functor is two actions plus two properties — namely, that the action on morphisms respects identities and composition —, a natural isomorphism can be thought as two NTs going in opposite directions, plus the assurance that their composites are identities; the downcased squares for the $(\times B) \dashv (B\to)$ and $L \dashv R$ adjunctions in the Figures 6a and 6b can be considered as being (two-dimensional) downcased names themselves (for natural isomorphisms), having the same meaning as their "linearized" versions,

$$(a^{\mathrm{op}}; c) \xrightarrow{\bullet} ((a, b \mapsto c) \leftrightarrow (a \mapsto (b \mapsto c)))$$
$$(a^{\mathrm{op}}; b) \xrightarrow{\bullet} ((a^L \mapsto c) \leftrightarrow (a \mapsto b^R))$$

and those meanings can either be the 6-uples

$$(\mathbf{Set}, \mathbf{Set}, (\times B), (B\to), \mathrm{uncur}, \mathrm{cur})$$
$$(\mathbf{A}, \mathbf{B}, L, R, \flat, \sharp)$$

or longer tuples including, say, the properties and the unit and the counit of the adjunction. The possibility of changing some definitions while keeping

the notations the same will be very important in sections 19 and 21, where that will be used to "project out" from the definitions all components which involve equalities of morphisms, keeping only the constructions.

10. Transmission

A good way to understand how *reconstruction* works is to think on a simpler, more extreme case. When I reconstruct something that I have half-forgotten, I do have vague memories about it... how do they act? On the other hand, if I am reconstructing something that I have received from someone else in an *incomplete*, but *reconstructible*, form, then the vague memories are out of the picture.

Consider the following diagram, which describes, in a simplistic way, how a theorem T, discovered by an author A and published in a paper P, is read and understood by a reader R. In the diagram the time flows to the left, and knowledge flows (roughly) downwards.

$$\begin{array}{rl} A_0 \longrightarrow & A_2 \\ T_1 \longrightarrow & T_2 \\ & T_3 \\ & P \\ & T'_0 \longrightarrow T'_1 \\ & R_0 \longrightarrow R_1 \end{array}$$

The theorem was discovered in the form T_1, but in order to make it transmissible the author changed it a bit, and it became T_2, then T_3 in written form, which was what got published. The author was also slightly transformed in the process, and at the time of the publication he had become A_2.

That particular theorem was difficult to state, so the reader R started (as R_0) by understanding just parts of its statement; let's call that preliminary object understood by our reader a "statement with holes". Then, through weeks of study, more and more of the theorem's statement, and of the proof of the theorem (let's use the term "theorem" to refer to both the statement and the proof), became clear to R, up to the point where the object in R's mind has become T'_1, which no longer has any holes; all gaps have been filled, the statement is now clearly a consequence of its proof, and our reader R, who at this point has become a slightly different person, R_1, now has even some *intuition* about the theorem (whatever that means)...

Our reader R has *reconstructed*, in his mind, the author A's theorem, from what was published in the paper P. R's task was harder than the task I face when I try to reconstruct a theorem that at some point I knew — if I review all that I have studied before, I *should* be able to fill up all my gaps, but R has no guarantee that just by looking in the literature he will be able to find all the missing knowledge he needs... even though we *expect* published papers to be clear enough, and complete enough.

The process of reconstruction performed by R *ought* to be considered an algorithm — even though we know that what R did to understand the paper P was, at best, a "method". Let's see why, by examining an ideal case.

Suppose that the A wanted to be clear, that he was lucid in his choice of exposition, and that the paper P was submitted to a journal with no space constraints. So A, and also his editors and referees, *want* the paper P to be as readable as possible. The reader R_0 receives the paper P being aware that the theorem T_3 in it should be readable, and starts to devote his time to understand T_3. After a few weeks R succeeds.

What did the reader R do? And what did A and the journal's editors do to be sure that A would succeed in his task?

When R checks the paper's details and fills up the gaps his process is akin to *proof search*, and thus a kind of *lifting*... What R does is too complex to be formalizable as an "algorithm"; yet the author and the editors are aware of what their intended readers are expected to be able to do, and they added to the paper enough "hints" to let the readers succeed — so the reader performs a "proof search with hints". There are algorithms that do proof searches with hints, so let's commit an abuse of language here and take the analogy with algorithms seriously: the reader R is performing a proof search with hints, and A provided enough hints in the paper P to be sure that, given P as input, the reader R will check its theorems completing all the missing details, then stop.

Let's now suppose that both A and R are people whose thinking is mostly diagrammatical, like the semi-fictional "me". How does does R (re)construct in his mind some "intuition" about the theorems? The theorem T_3 was published in algebraic form, so part of R's task — and he needs to do this to be able to fill the logical gaps — is to find the diagrams to let him reason about the paper; another part of his task is to work out the details in the paper's examples; for that he needs to take his "general" diagrams and apply them to particular cases; this is *specialization*, as in Section 2. What we call "intuition" should comprise the ability to specialize, plus a lot more.

11. Intuition

When R finally understands the paper P he would have developed some "intuitions" about the theorem T. His intuitions may or may not be the same as A's, so let's name them differently: I_A for the author's intuitions, I_R for the reader's.

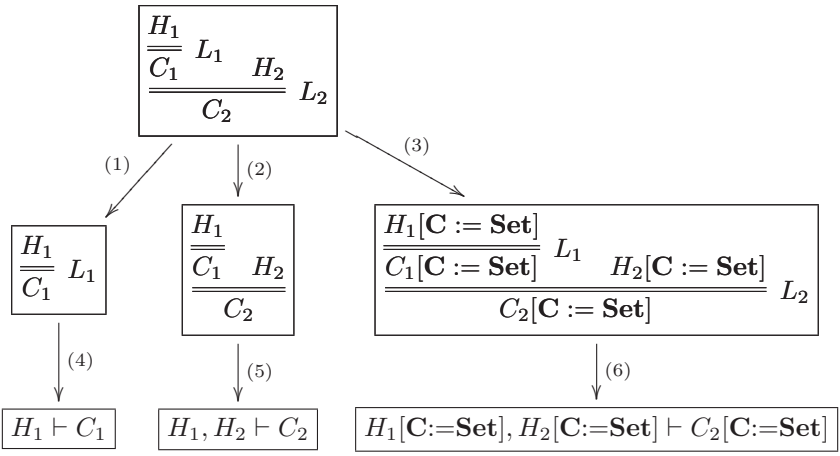
Instead of taking the easy way and saying that "it is impossible to talk about what mathematical intuition is", we will improvise a simple model that will let us talk about *some aspects* of having intuition about a theorem. Let's suppose that that theorem T says that when the hypotheses H_1 and H_2 hold, then the conclusion C_2 also does; and the proof of T is made of two lemmas,

L_1 and L_2, and structured like this:

$$\dfrac{\dfrac{H_1}{\overline{\overline{C_1}}\ L_1} \qquad H_2}{\overline{\overline{C_2}}\ L_2}$$

so the stronger theorem $H_1 \wedge H_2 \vdash C_1 \wedge C_2$ is also true. Note that the tree above represents T_1', not T_3; T_3 was written in a "linear", "algebraic" way, to comply with the usual mathematical practice, even though both A and R tend to think diagrammatically.

The reader R_1 is able to do several things with T_1'; most of them can be understood diagrammatically, as movements through parts of the diagram below. Let's use the following convention (that will hold for this diagram only): a double bar with the name of a lemma at its right will stand for *all the steps in the lemma* — they are known, but are not visible —, while a double bar without a label will mean a situation where the intermediate steps are not known at that moment, and will have to be reconstructed.



The center box is a sketch of the full proof, and the small box below it is the statement of the full theorem. If the reader R_1 remembers any of those, he can reconstruct the other one by projecting or lifting through (5). The box at the top is the full proof, and R_1 can reconstruct it by completing the sketch of the proof, lifting through (2).

One day R_1 decides to present the theorem to some colleagues. He assigns a temporal order to the lemmas: "Lemma L_1 has to be presented first". Its statement and its proof are obtained by projecting through (1) and (4). Note that (1) is a case of *zooming in* into a part of the proof.

R_1 is also able to *use* the theorem. He can apply it to a particular case, projecting through (3) and (6) or just through (3) (compare that with the arrow (3) in Figure 8). He can also reuse the *structure* of parts of the proof, but changing the theorem in deeper ways (e.g., in the Curry-Howard isomorphism); this is not shown above.

We will pretend that "having intuition about a theorem" means having the abilities discussed above: remembering parts, reconstructing, zooming out, zooming it, temporal order, transmission, specializing, reusing the structure.

12. Skeletons of proofs

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.

Now the irresistible questions are: which kinds of objects do have skeletons? What do these skeletons look like? How does the reconstruction process work, and how much of it can be performed by computers? When is it that the liftings become ambiguous, what kinds of hints are needed, and how should we specify them? And in what situations is this idea doomed to fail, because for each non-trivial way of separating the object's data into "skeleton" and "non-skeleton" something doesn't work?

Answering all this in a general setting is obviously a daunting task. A first natural step, though, is to start from a handful of natural examples — and then say: these are our archetypal examples of skeletons, projections, and liftings. How do we formalize and generalize what we got here?

In Section 19 we will sketch an approach that *may* yield a reasonbably rich family of examples: namely, that on a sizeable fragment of Category Theory all definitions can be split into a *structure* part plus *properties*, and each theorem into a *construction* plus an *equational part*. A big part of Mathematics is definitions plus theorems — and in that fragment of Category Theory these can be clearly split into a "syntactical" skeleton, with just the "structures" and "constructions", plus, on top of that, a reconstructible, "equational" flesh. We will call the system with just this skeleton the "syntactical world".

It would be very hard to explain precisely the general ideas here before first showing a language on which they can make sense, so we will now look at some examples. We will have to use hyperdoctrines, even though they are weaker, less familiar, and harder to define than toposes; that's because of technical difficulties that we will discuss in section 20.

13. Hyperdoctrines

Let's take the *beginning* of the definition of a hyperdoctrine ([Law69], [Law70], [See83], [Tay86], [Jac99]). A hyperdoctrine is a cloven fibration $p : \mathbb{E} \to \mathbb{B}$, where the "base category" $\mathbb{B}$ has finite products and each fiber $\mathbb{E}_B$ is cartesian closed, plus for each morphism $f : A \to B$ in $\mathbb{B}$ adjoints $\Sigma_f \dashv f^* \dashv \Pi_f$ for

the change-of-base functor f^*, *plus more structure*; but let's skip the "plus more structure" part for the moment — we will describe that extra structure briefly at Section 14, and the full details can be found at [Och10].

It turns out that this definition *generalizes* a familiar object: the "codomain fibration", $\mathrm{Cod} : \mathbf{Set}^{\rightarrowtail} \to \mathbf{Set}$ — that we will abbreviate as Pred — is a hyperdoctrine. Actually Pred can be considered to be the *archetypical* hyperdoctrine, in a sense that can be made precise; we will return to "generalizations" and "archetypes" in Section 16.

An object P of $\mathbf{Set}^{\rightarrowtail}$ is a monic: $P \equiv (A' \rightarrowtail A)$. A morphism $P \to Q$ in $\mathbf{Set}^{\rightarrowtail}$, where $Q \equiv (B' \rightarrowtail B)$, is a pair of arrows $(f' : A' \to B', f : A \to B)$ making the obvious square commute. The projection functor Cod takes $P \equiv (A' \rightarrowtail A)$ to A and $(f', f) : P \to Q$ to f.

A monic with codomain A, $(A' \rightarrowtail A)$, is said to be a *subobject* of A. In $\mathbf{Set}$ we have *canonical subobjects* — the ones whose maps are inclusions — and we can think of them as being *predicates* over sets. We will have a special shorthand for predicates: instead of writing, for example,

$$\{\,(x,y) \in X \times Y \mid P(x,y)\,\} \hookrightarrow X \times Y$$

we will write just:

$$\{\,x, y \parallel Pxy\,\}$$

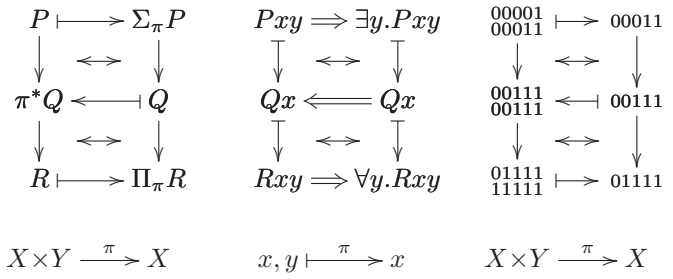
The double bar in '$\{\,x, y \parallel Pxy\,\}$' is to remind us that this is two objects, plus a map; and we write the '$\rightarrowtail$' as '$\hookrightarrow$' when we want to stress that the map is an inclusion.

We will draw objects of $\mathbb{E}$ above their projections, and we'll usually omit the projection arrows. We will downcase a predicate like '$\{\,x, y \parallel Pxy\,\}$' in diagrams as just '$P(x,y)$' — the '$x, y$' part can be recovered by looking down.

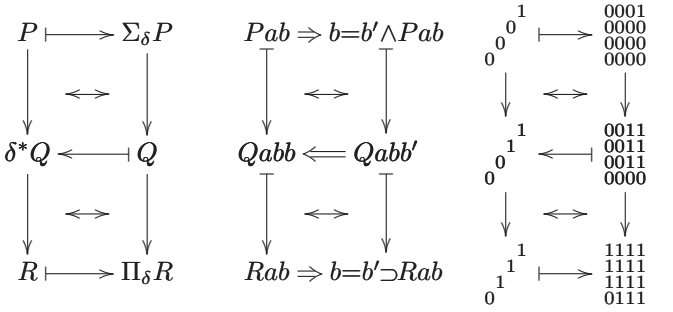
Let's focus on what happens in Pred. For a map $f : A \to B$ in $\mathbb{B}$, the change-of-base functor f^* takes each predicate $\{\,b \parallel P(b)\,\}$ over B to a predicate $\{\,a \parallel P(f(a))\,\}$ over A. When f is a projection map, like $\pi : X \times Y \to X$, the adjoints Σ_π and Π_π "are" $\exists y$ and $\forall y$ (Figure 7a); when f is the "diagonal" map $\Delta : B \to B \times B$ or the "dependent diagonal" $\delta : A \times B \to A \times B \times B$, the adjoints "are" the operations '$b{=}b'\wedge$' and '$b{=}b'\supset$' (Figure 7b). This is not hard to believe if we start with the right examples, and we check first the particular cases and then generalize. Take $X = \{0,1,2,3,4\}$, $Y = \{3,4\}$, $A = \{0\}$, $B = \{0,1,2,3\}$, and let's use a positional notation for predicates: we will write $\{\,x, y \parallel x \geq y\,\}$ as $\begin{smallmatrix}00001\\00011\end{smallmatrix}$, i.e.:

$$\{\quad \begin{smallmatrix}(4,4),\\(3,3),(4,3)\}\end{smallmatrix} \quad\hookrightarrow\quad \begin{smallmatrix}\{(0,4),(1,4),(2,4),(3,4),(4,4),\\(0,3),(1,3),(2,3),(3,3),(4,3)\}\end{smallmatrix}$$

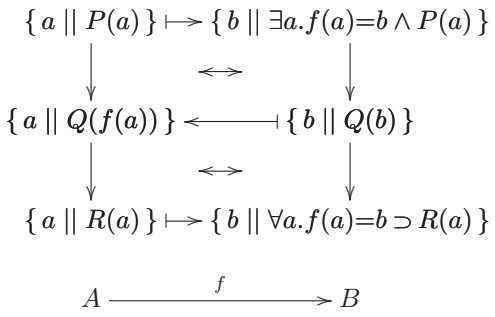
In Figures 7a and 7b the left side shows the abstract view, the right side shows a very particular case, and in the middle, in downcased notation, we see how these change-of-base functors and their adjoints act on arbitrary predicates. Also, each arrow '$\leftrightarrow$' in an adjunction square stands for two transpositions, one in each direction. Let's establish another convention: $P \to f^*Q$ is an hom-set, but $P \vdash f^*Q$ is (the name of) a morphism — we have



(A) Adjoints to π^* (quantifiers).



(B) Adjoints to δ^* (equality).



(C) Adjoints to an arbitrary change-of-base functor.

FIGURE 7. Adjoints to change-of-base in a hyperdoctrine.

$P \vdash f^*Q : P \to f^*Q$. With that convention we can write the transpositions rules (cf. the "mate rules" in [Jac99]) as:

$$\frac{Q \vdash \Pi_\pi R}{\pi^* Q \vdash R}\ \forall^\flat \qquad \frac{x; Qx \vdash \forall y.Rxy}{x, y; Qx \vdash Rxy}\ \forall^\flat$$

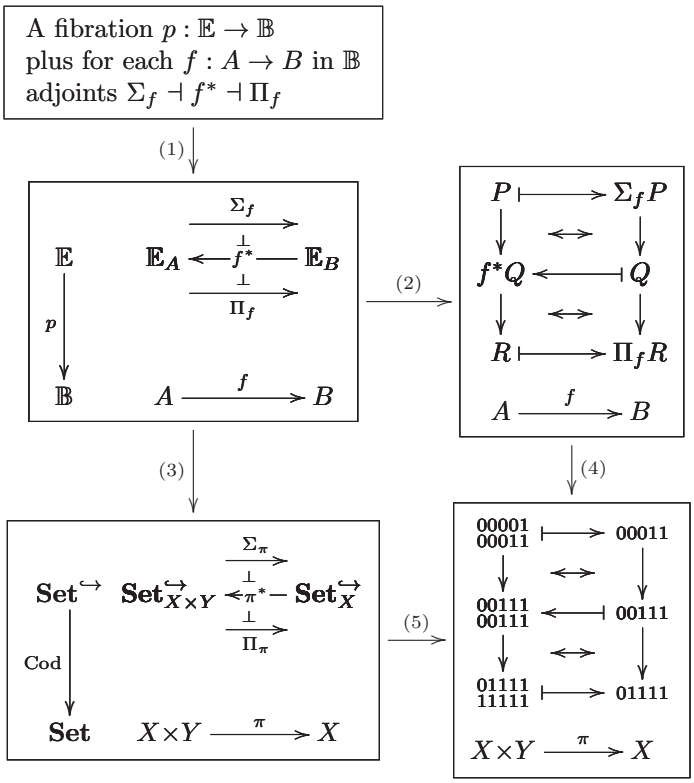


FIGURE 8. How the hyperdoctrine diagrams in Figure 7 were obtained.

$$\frac{\pi^* Q \vdash R}{Q \vdash \Pi_\pi R}\ \forall^\sharp \qquad \frac{x, y; Qx \vdash Rxy}{x; Qx \vdash \forall y. Rxy}\ \forall^\sharp$$

Note that '$\vdash$' has different meanings in standard and in downcased notations. We will only use '$\vdash$' to refer to vertical morphisms.

It turns out that it is possible to *define* the adjoints $\Sigma_f \dashv f^* \dashv \Pi_f$ in Pred, for an arbitrary $f : A \to B$, from just f^*, Σ_π, Π_π, Σ_δ, Π_δ. The transpositions and the functor actions implicit in the adjunction diagram in Figure 7c above can be expressed as derived rules. The proof is hard (see [See83]); we will see how to understand it diagrammatically in [Och10].

What really matters here is: how were the diagrams above obtained?

Look at Figure 8; it shows some of the projections involved.

We start with the definition in English, at the top. By representing it diagrammatically (projection (1)) we lose the subtleties of the English language — see [Fre76] and [FS90] for a diagrammatic notation in which at

least the quantifiers are preserved — but we add positional notation. The projections (3) and (4) are specializations: we take $\mathbb{E} := \mathbf{Set}^{\hookrightarrow}$, $p := \mathrm{Cod}$, $A := X \times Y$, and so on. The arrows (2) and (5) produce internal diagrams, as in the section 9. The downcased diagrams are not shown; Figure 8 is a map of interesting projections, and it could have been far larger than it is.

14. Preservations, Frobenius, Beck-Chevalley

The complete definition of a hyperdoctrine is what we have seen in the last section, plus these *properties*:

- each f^* preserves, modulo iso, the $\top$, the $\wedge$, and the $\supset$ of each fiber;
- the Frobenius Property holds,
- the Beck-Chevalley Condition holds.

We can regard these properties as *structure*. They can all be stated in the same way: for $f : A \to B$ in $\mathbb{B}$ and for predicates P and Q over B, we have natural constructions $\mathsf{P}\top^\natural$, $\mathsf{P}\wedge^\natural$, $\mathsf{P}\supset^\natural$, $\mathsf{Frob}^\natural$, that produce arrows that we want to be isos; so we establish rules $\mathsf{P}\top$, $\mathsf{P}\wedge$, $\mathsf{P}\supset$, Frob, that produce arrows going in the opposite directions of the previous ones.

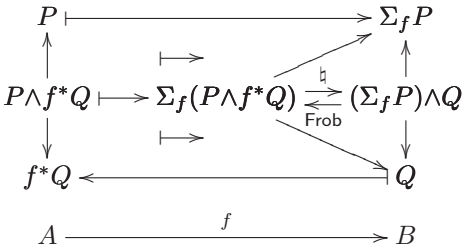
$$\frac{f : A \to B}{f^*\top_B \vdash \top_A}\ \mathsf{P}\top^\natural \qquad \frac{f : A \to B}{\top_A \vdash f^*\top_B}\ \mathsf{P}\top$$

$$\frac{f \quad P \quad Q}{f^*(P\wedge Q) \vdash f^*P \wedge f^*Q}\ \mathsf{P}\wedge^\natural \qquad \frac{f \quad P \quad Q}{f^*P \wedge f^*Q \vdash f^*(P\wedge Q)}\ \mathsf{P}\wedge$$

$$\frac{f \quad Q \quad R}{f^*P \wedge f^*Q \vdash f^*(Q\supset R)}\ \mathsf{P}\supset^\natural \qquad \frac{f \quad Q \quad R}{f^*(Q\supset R) \vdash f^*P \supset f^*Q}\ \mathsf{P}\supset$$

$$\frac{f \quad P \quad Q}{\Sigma_f(P\wedge f^*Q) \vdash (\Sigma_f P)\wedge Q}\ \mathsf{Frob}^\natural \qquad \frac{f \quad P \quad Q}{\Sigma_f(P\wedge f^*Q) \vdash (\Sigma_f P)\wedge Q}\ \mathsf{Frob}$$

The natural construction for the $\mathsf{Frob}^\natural$ arrow is given by the diagram below. Note that it shows both $\mathsf{Frob}^\natural$ (going rightwards, shortened to just '$\natural$') and Frob:



Similarly to what we have done in the section 4, we can extract from the diagram above a dictionary defining some names in terms of the others; note that due to our conventions $P \wedge f^*Q \to P$ is a hom-set, but $P \wedge f^*Q \vdash P$ is a particular morphism, and we can treat it as a name. By expanding the

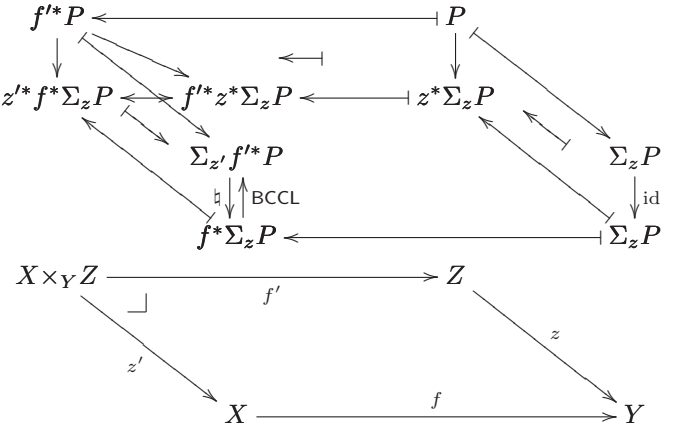
definitions in $\Sigma_f(P \wedge f^*Q) \vdash (\Sigma_f P) \wedge Q$, we obtain a definition for the $\mathsf{Frob}^\natural$ rule:

$$\frac{f \quad P \quad Q}{\Sigma_f(P \wedge f^*Q) \vdash (\Sigma_f P) \wedge Q}\ \mathsf{Frob}^\natural \quad :=$$

$$\dfrac{\dfrac{\dfrac{P \quad \dfrac{f \quad Q}{f^*Q}\ \text{c.o.b.}}{P \wedge f^*Q \vdash P}\ \pi \qquad f}{\Sigma_f(P \wedge f^*Q) \vdash \Sigma_f P}\ \Sigma_f \qquad \dfrac{\dfrac{P \quad \dfrac{f \quad Q}{f^*Q}\ \text{c.o.b.}}{P \wedge f^*Q \vdash f^*Q}\ \pi' \qquad \Sigma_f{}^\flat}{\Sigma_f(P \wedge f^*Q) \vdash Q}}{\Sigma_f(P \wedge f^*Q) \vdash (\Sigma_f P) \wedge Q}\ \langle , \rangle$$

The diagram also says that the arrow generated by the Frob rule is inverse to the arrow generated by the (derived) rule $\mathsf{Frob}^\natural$. This is by a convention: in a '$\leftrightarrows$' the two arrows are the two directions of an iso.

The statement of the Beck-Chevalley Condition ("BCC" from now on) is slightly more complex: it requires four arrows in $\mathbb{B}$. For any commutative square in $\mathbb{B}$ as below,



and any object P over Z, we have a natural construction for an arrow:

$$\mathsf{BCCL}^\natural : \Sigma_{z'} f'^* P \to f^* \Sigma_z P$$

The rule BCCL says that when f, f', z, z' form a pullback the arrow $\mathsf{BCCL}^\natural$ has an inverse, BCCL:

$$\frac{f \quad z \quad f' \quad z' \quad P}{f^* \Sigma_z P \vdash \Sigma_{z'} f'^* P}\ \mathsf{BCCL}$$

We will sometimes write $\mathsf{BCCL}^\natural$ and BCCL together as an iso, and call that iso BCCL (by a slight abuse of language); and we simplify the diagram for Beck-Chevalley by drawing only its square in $\mathbb{B}$ and on top of it the two faces of the cube in $\mathbb{E}$ that border on the missing edge. We will sometimes draw the arrows generated by $\mathsf{P}\top$, $\mathsf{P}\wedge$, $\mathsf{P}\supset$ and Frob as isos, too.

15. The Eq-Elim rule

The main interest of hyperdoctrines is that they are exactly the categories in which we can interpret (a certain system of typed, intuitionistic) first-order logic. The proof of this includes a way to interpret each deduction rule of first-order logic categorically, and another "translation" going in the opposite direction.

Let's look at a tiny part — possibly the hardest one — of these translations.

The rule for equality-elimination in Natural Deduction can be formulated as this ([See83], p.507):

$$\frac{b{=}b' \quad Qabb}{Qabb'} =E$$

Categorically, it will become this morphism (we will call it $=E$):

$$\{\, a, b, b' \mid\mid b{=}b' \wedge Qabb \,\} \vdash \{\, a, b, b' \mid\mid Qabb' \,\}$$

There are three different natural constructions in a hyperdoctrine for objects deserving the name $\{\, a, b, b' \mid\mid b{=}b' \wedge Qabb \,\}$. We will use this one in $=E$:

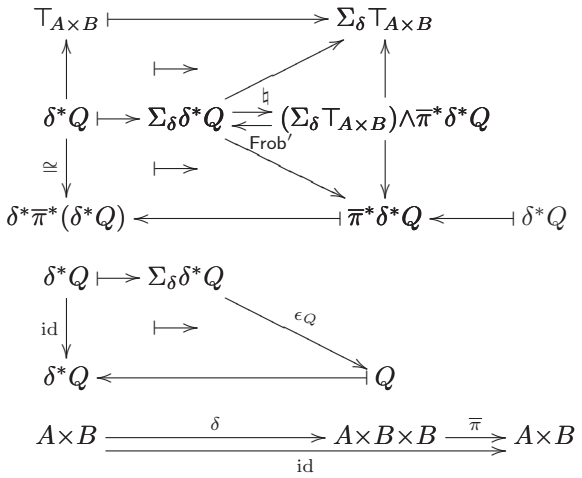
$$\dfrac{\dfrac{a, b, b' \mapsto b, b' \quad \dfrac{b \mapsto b, b' \quad \{\, b \mid\mid \top \,\}}{\{\, b, b' \mid\mid b{=}b' \,\}}\,\Sigma_0\text{ c.o.b.}}{\{\, a, b, b' \mid\mid b{=}b' \,\}}\text{ c.o.b.} \quad \dfrac{a, b \mapsto a, b, b' \quad \dfrac{a, b \mapsto a, b, b' \quad \{\, a, b, b' \mid\mid Qabb' \,\}}{\{\, a, b \mid\mid Qabb \,\}}\text{ c.o.b.}}{\{\, a, b, b' \mid\mid Qabb' \,\}}\text{ c.o.b.}}{\{\, a, b, b' \mid\mid b{=}b' \wedge Qabb' \,\}}\,\wedge$$

$$\dfrac{\dfrac{\overline{\pi}' \quad \dfrac{\Delta \quad \top_B}{\Sigma_\Delta \top_B}\,\Sigma_0\text{ c.o.b.}}{\overline{\pi}'^{*}\Sigma_\Delta \top_B}\text{ c.o.b.} \quad \dfrac{\overline{\pi} \quad \dfrac{\delta \quad Q}{\delta^{*}Q}\text{ c.o.b.}}{\overline{\pi}^{*}\delta^{*}Q}\text{ c.o.b.}}{\overline{\pi}'^{*}\Sigma_\Delta \top_B \wedge \overline{\pi}^{*}\delta^{*}Q}\,\wedge$$

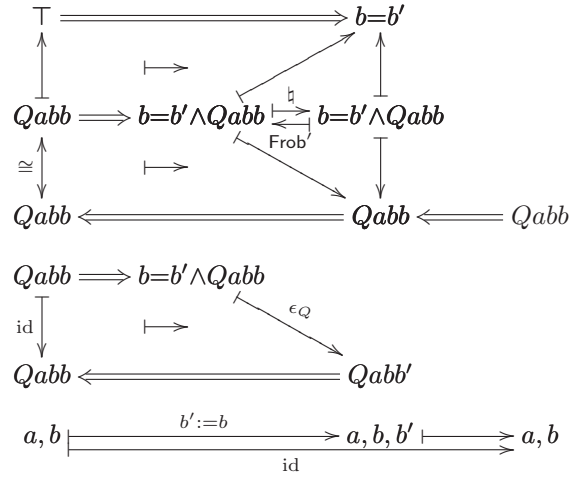
Note that $\overline{\pi} : A{\times}B{\times}B \to A{\times}B$ is the projection on the first two coordinates, and $\overline{\pi}' : A{\times}B{\times}B \to B{\times}B$ the projection on the last two coordinates. To show that the rule $=E$ can be interpreted in a hyperdoctrine we need a way to construct, for arbitrary objects A, B in the base category and an arbitrary predicate Q over $A{\times}B{\times}B$, a morphism $\overline{\pi}'^{*}\Sigma_\Delta \top_B \wedge \overline{\pi}^{*}\delta^{*}Q \vdash Q$. We will construct it as a composite of three maps.

Take $P := \top_B = \{\, b \mid\mid \top \,\}$, $f := \overline{\pi}^{*}$, $z := \Delta$, $z' := \delta$, $f' := \pi'$ in the diagram for Beck-Chevalley. Then the BCCL iso says that *two* different categorical constructions for $\{\, a, b, b' \mid\mid b = b' \,\}$ are isomorphic, and by drawing also the P$\top$ iso and its image by Σ_δ we get isos between the *three* different categorical constructions for $\{\, a, b, b' \mid\mid b = b' \,\}$:

Now look at the diagram in Figure 9a. The arrow $\mathsf{Frob}'^{\natural}$ (going rightwards, shortened to $\natural$) is built in the same way as $\mathsf{Frob}^{\natural}$, but using the maps ! and $\cong$ in place of π and π'. In any fibration we have natural isos $f^{*}g^{*}P \leftrightarrow (f;g)^{*}P$ and $\mathrm{id}^{*}P \leftrightarrow P$ (for *diagrammatic* proofs for that, see [Och10]), and as $\delta;\overline{\pi} = \mathrm{id}$, this gives us the map $\cong$. The arrow Frob' is



(A) The components 'Frob'' and ϵ_Q of the $(=E)$ map (standard)

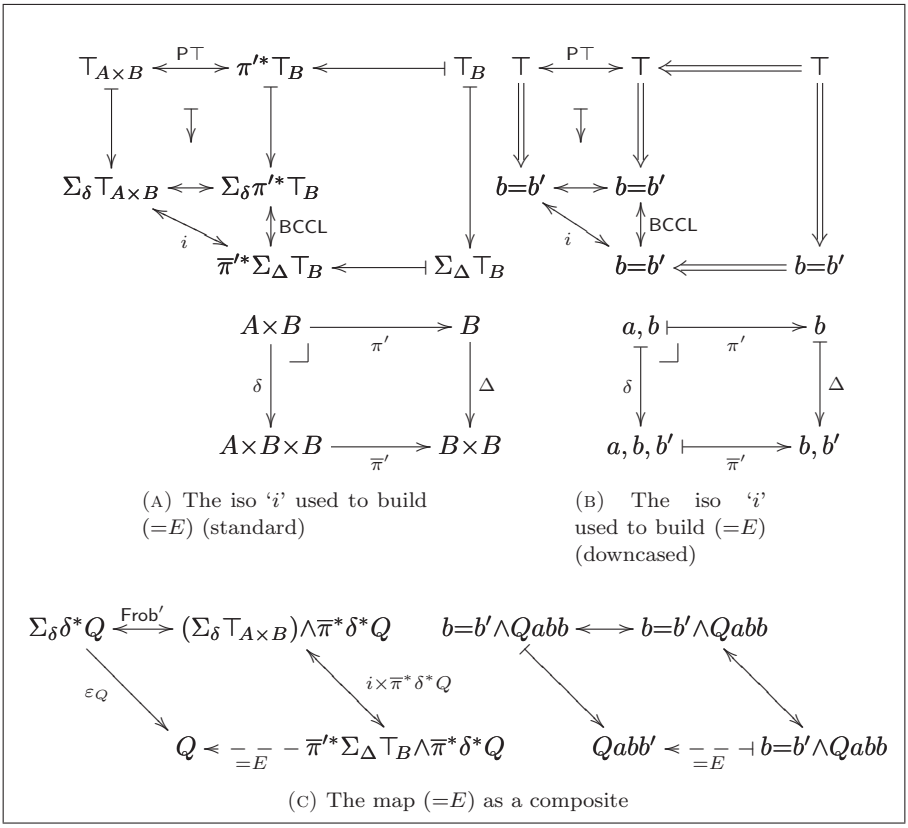


(B) The components 'Frob'' and ϵ_Q of the $(=E)$ map (downcased)

FIGURE 9. $(=E)$ via Frobenius

the inverse of $\mathsf{Frob}'^{\natural}$; its construction (not shown) is an easy consequence of Frobenius. The map ε_Q is a counit for the adjunction $\Sigma_\delta \dashv \delta^*$.

If $Q \equiv \{\, a, b, b' \mid\mid Qabb' \,\}$ then we can downcase the diagram above into the one in Figure 9b; the arrow $=E$ is the composite in Figure 10c.



(A) The iso 'i' used to build $(=E)$ (standard)

(B) The iso 'i' used to build $(=E)$ (downcased)

(C) The map $(=E)$ as a composite

FIGURE 10. $(=E)$ as a composite

16. Archetypal Models

Imagine that we have placed side to side the downcased and the standard diagrams of the last section.

When we go from downcased to standard — e.g, from $\{\, a, b \parallel b{=}b' \wedge Qabb \,\}$ to $\Sigma_\delta \delta^* Q$ — we are attributing a precise meaning to a (potentially ambiguous) "name"; however, when we go in the opposite direction, from standard to downcased, we are "attributing meaning" in another sense: we are giving an "intuitive meaning" (or better: "intuitive content") to something that, if we had received it in an article, as in the section 10, could have felt initially as something purely abstract...

I had to make some "informal definitions" in the course of this paper, using terms that I could not define precisely, like *diagrammatical reasoning*, *reconstruction*, and *intuition*... I will have to make a few more.

> The *archetypal model* for a structure — for example, the archetypal model for a hyperdoctrine, which is going to be Pred — is

a particular case of that structure that "suggests" a certain "language" for working on that structure — i.e., for doing constructions and proofs on it.

That "archetypical language" does not need to be unambiguous — think in $\{\, a, b \mid\mid b{=}b' \wedge Qabb \,\}$ and its several different precise meanings —, does not need to have a downcased version — in section 13 the internal diagrams were what mattered, the downcasings were mentioned just in passing —, and does not need to be convenient for expressing *all* possible constructions. What is relevant is that the archetypical language, *when used side-to-side with the "algebraic" language*, should give us a way to reason, both *intuitively* and *precisely*, about the structure we're working on; in particular, it should let us formulate reasonable conjectures quickly, and check them with reasonable ease... but what are "reasonable conjectures", and where do they come from?

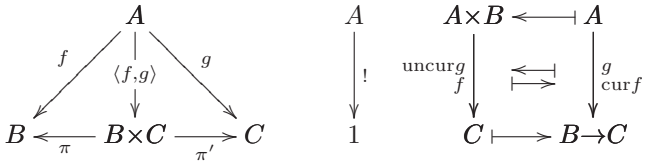
If we want to be able to reconstruct a theory from minimal information we need to have ways to: 1) generate "reasonable conjectures", 2) filter out those which are either impossible or too hard to prove, 3) prove the others. *Bounded* proof search takes care of points (1) and (2); to give a partial answer to (1) we will concentrate our attention on Category Theory — more specifically, on situations where we are trying to generalize a "base case". There we have two (non-disjoint) sources of "reasonable conjectures": 1a) constructions and proofs that make sense and hold in the base case, that we may expect to generalize; 1b) *language*. Our choice of names for objects in a hyperdoctrine gave us several different formal meanings for $\{\, a, b \mid\mid b{=}b' \wedge Qabb \,\}$ — we expect them to be at least equivalent modulo isomorphism. This is similar to writing $a + b + c$ instead of $a + (b + c)$ or $(a + b) + c$: our choice of language "suggests" that $a + (b + c) = (a + b) + c$.

An "archetypal example", in which all the main ideas appear, is:

Set is the archetypical CCC.

17. Cartesian Closed Categories

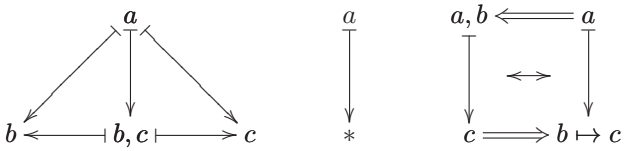
A *Cartesian Closed Category* $(\mathbf{C}, \times, 1, \to)$ is a category $\mathbf{C}$ plus a "cartesian closed structure" $(\times, 1, \to)$ on it. The following diagram fixes the (categorical) notation that we will use for the operations induced by $(\times, 1, \to)$:



Each of its three parts can be attributed a precise meaning, as we did with the adjunction square in section 9; and if we regard each of them as a different adjunction (see [Awo06], pages 182 and 188, for the three adjunctions), then we get the equations that these operations have to obey.

The Big Theorem is: *CCCs are exactly the categorical models for the simply-typed λ-calculus with pairs and unit.*

The precise, "standard" statement for that theorem, including definitions, statements, and proofs, is quite long. Amazingly, it can be reconstructed from just this downcased diagram,



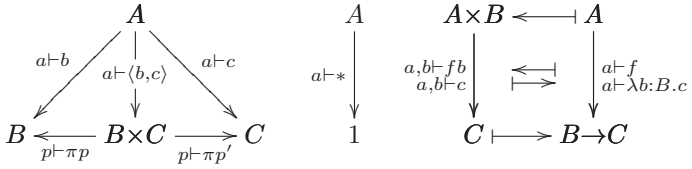
plus this downcasing of the Natural Deduction rules for ∧, ⊤, and ⊃:

$$\frac{\begin{array}{cc} a & a \\ \vdots & \vdots \\ b & c \end{array}}{b,c}\ \langle , \rangle \qquad \frac{b,c}{b}\ \pi \qquad \frac{b,c}{c}\ \pi'$$

$$\frac{a}{*}\ !$$

$$\frac{\begin{array}{cc} a & a \\ \vdots & \vdots \\ b & b \mapsto c \end{array}}{c}\ \text{app} \qquad \frac{\begin{array}{c} a \\ \vdots \\ c \end{array}}{b \mapsto c}\ \lambda; 1$$

Starting from that, we uppercase the downcased diagram in the following way, that makes clear how an intermediate, "sequent-like" form (as in [LS86], pp.47–49), should behave:



Then the next steps are to state precisely:

- the syntax and the rules of the type system,
- the operations of a CCC,
- the translation of each of the rules of the type system,
- the translation of each of the operations of the CCC,
- the reductions and conversions of the type system,
- the equations obeyed by the operations of a CCC,

and then we have to prove that the CCC equations are respected by the translation to λ-calculus, and that the λ-calculus equations are respected by the translation to categories.

We also need to allow "impurities" in both the λ-calculus and in the CCCs. Take a look at the CCC of the next section: it has extra *constants* — an object A and morphisms $\ulcorner 0 \urcorner$, $\ulcorner 1 \urcorner$, $+$, $\cdot$ — that do not exist in all CCCs, and these constants obey certain *equations*. In a *Pure Type System* ([Geu93]) extra constants and equations are not allowed, but in theorems like the Big Theorem above ([Jac99] has many more like it) we allow in the type system exactly the kinds of constants and equations that are easy to interpret in the categorical models. The kinds of allowed "impurities" are usually defined in the beginning, in the definition of the type system and of the categorical structure, but this can be delayed until after the translations are presented.

18. Objects of Line Type

Our approach also works in cases where we do have an archetypical language with clear intuitive content, but where we have no concrete archetypical model besides a term model built purely syntactically. Through this section let's pretend that we do not know Synthetic Differential Geometry ([Koc80], [Bel08], [MR91]).

Let $(A, \ulcorner 0 \urcorner, \ulcorner 1 \urcorner, +, \cdot)$ be a commutative ring in a CCC. That means: we have a diagram

$$1 \underset{\ulcorner 1 \urcorner}{\overset{\ulcorner 0 \urcorner}{\rightrightarrows}} A \underset{\cdot}{\overset{+}{\leftleftarrows}} A \times A$$

$$\begin{aligned} * &\longmapsto 0 \\ * &\longmapsto 1 \\ a+b &\longmapsto a, b \\ ab &\longmapsto a, b \end{aligned}$$

and the morphisms $\ulcorner 0 \urcorner$, $\ulcorner 1 \urcorner$, $+$, $\cdot$ behave as expected.

Let D be the set of zero-square infinitesimals of A, i.e., $\{ \varepsilon \in A \mid \varepsilon^2 = 0 \}$; D can be defined categorically as an equalizer. If we take $A := \mathbb{R}$, then $D = \{0\}$; but if we let A be a ring with nilpotent infinitesimals, then $\{0\} \subsetneq A$.

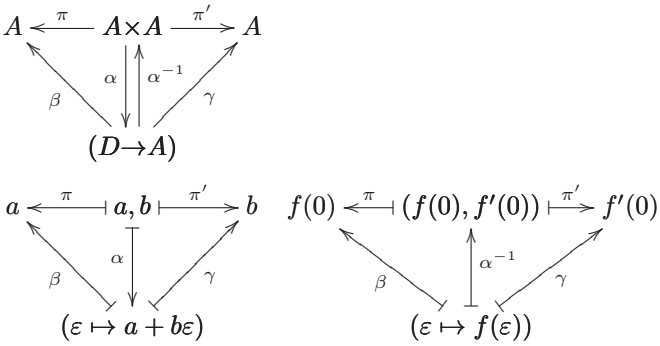
The main theorem of [Koc77] says that if the map

$$\begin{array}{rccc} \alpha: & A \times A & \to & (D \to A) \\ & (a, b) & \mapsto & \lambda \varepsilon{:}D.(a + b\varepsilon) \end{array}$$

is invertible, then we can use α and α^{-1} to *define* the derivative of maps from A to A — *every* morphism $f : A \to A$ in the category $\mathbf{C}$ will be "differentiable" —, and the resulting differentiation operation $f \mapsto f'$ behaves as expected: we have, for example, $(fg)' = f'g + fg'$ and $(f \circ g)' = (f' \circ g)g'$.

Commutative rings with the property that their map α is invertible are called *ring objects of line type*. ROLTs are hard to construct, so most of the proofs about them have to be done in a very abstract setting. However, if we can use the following downcasings for α and α^{-1} — note that $\beta = (\alpha^{-1}; \pi)$,

that $\gamma = (\alpha^{-1}; \pi')$, and that these notations do not make immediately obvious that α and α^{-1} are inverses —,



then all the proofs in the first two sections of [Koc77] can be reconstructed from half-diagrammatic, half-λ-calculus-style proofs, done in the archetypal language, where the intuitive content is clear. This will be shown in a sequel to [Och10].

19. The syntactical world

We can now explain our archetypal projection: the one from the "real world" to the "syntactical world" for a fragment of Category Theory, that we mentioned in section 12.

We have been avoiding all mentions to equations between morphisms — for example, in the section 9 we glossed over the usual standard requirement that $f^{\sharp\flat} = f$. That was deliberate.

A category $\mathbf{C}$ is a 7-uple,

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

where the three last components are assurances that the composition $\circ_{\mathbf{C}}$ is associative and that the identities behave as expected with respect to composition at the left and the right. These three last components are exactly the ones whose typings — all this can be formalized in an adequate type system — involve equalities of morphisms. By dropping them we get what we will call the *proto-category* associated to $\mathbf{C}$:

$$\mathbf{C}^- = (\mathbf{C}_0, \mathrm{Hom}_{\mathbf{C}}, \mathrm{id}_{\mathbf{C}}, \circ_{\mathbf{C}})$$

The same idea can be applied to lots of categorical structures. We can define, in a similar way, proto-functors, proto-natural transformations, proto-isos, proto-adjunctions, proto-products, proto-terminals, proto-exponentials, proto-cartesian-closed categories, a proto-Yoneda lemma, proto-fibrations, and so on. Also, it turns out that many categorical proofs can be projected onto their corresponding "proto-proofs", by dropping all parts that involve

equalities of morphisms — and the resulting proto-proofs keep the *constructions* of the original proofs, but leave out the diagram chasings. This is explained in great detail, with many diagrams, at [Och10].

20. The problem with toposes

Elementary toposes are very simple to define — an elementary topos is just a CCC with pullbacks and a classifier object — and it is well-known that they are exactly the categorical models for a certain (intuitionistic) fragment of Set Theory (called "IST" in [Bel88]). *Why did we have to resort to clumsy hyperdoctrines?*

The problem is that we can't define *classifier object* without defining *monic arrow* first; all my attempts to find a usable definition of "proto-monic" have failed rather miserably[1], so I can't rely on either "proto-classifiers" or "proto-toposes". This means that if I had to stick with toposes I wouldn't be able to apply the idea of "projection into the syntactical world" to the metatheory, i.e., to the categorical models for the polymorphic type theory needed to formalize the projection from the real world into the syntactical world.

The usual approach to interpreting IST in a topos is to use the classifier object to define the logical connectives and the quantifiers, and then check that these definitions have all the expected properties. An alternative approach is to split this into two parts: 1) a classifier on a CCC with pullbacks induces a hyperdoctrine structure on it, and 2) the hyperdoctrine structure is *exactly* what is needed to interpret first-order logic with equality. This is how things are done in [Jac99], which then proceeds to show which extra structure is needed for interpreting polymorphism and dependent types... These are beautiful but hard theorems, whose details are not as widely-known as they should; what I felt was missing the most when I learned them was a way to make their intuitive content clearer. The tools in this paper *can* be a way to fill that gap — the next section sketches what that could mean.

21. Formalizing diagrams (in Type Theory)

We saw at sections 3 and 7 how to make each downcased "name" stand for both a "name" and its "meaning". In our downcased categorical diagrams, each node and each arrow has a definite meaning as a categorical entity; so, let's consider each node and arrow in a downcased diagram a *diagrammatical name*.

Note that diagrammatic names are *positional* — in the sense that entities with the same apparent names but at different positions of a diagram are allowed to have different meanings. That happened, for example, in Figure 9,

[1] The reader who doesn't believe that this can be hard is urged to try to find the syntactical skeleton of the proof that in a topos every singleton map $\{\cdot\}_A : A \to \Omega^A$ is monic.

where both $\delta^* Q$ and $\delta^* \overline{\pi}^* (\delta^* Q)$ became $\{\, a, b \mid\mid Qabb \,\}$ — abbreviated to a '$Qabb$' above an 'a, b' — in the downcasing.

Now take any downcased diagram D, and draw two *copies* of it, one above another, like this:

$$\begin{array}{c} D \\ \downarrow \\ D^- \end{array}$$

Let's take the positionality of names a step further. At the top diagram, D, all meanings are in the "real world", and are non-proto categorical entities. At the lower diagram, D^-, in the "syntactical world", each meaning is the proto-categorical entity corresponding to the non-proto entity above. For example, the meaning for an iso arrow '$a, b \leftrightarrow b, a$' will be just a pair $(\langle \pi'_{AB}, \pi_{AB} \rangle, \langle \pi'_{BA}, \pi_{BA} \rangle)$ in the syntactical world, but in the real world it will be this plus the assurances that both composites are identities.

Suppose that we tag with a different number — in light gray, say — each node and each arrow in the diagram $D \to D^-$; for example, our two copies of '$a, b \leftrightarrow b, a$' may get tagged as '$\leftrightarrow_{2099}$' in the real-world diagram, and as '$\leftrightarrow_{99}$' in the syntactical world. With this we get numerical suffixes that we can use for the corresponding terms, and in the formalization of that diagram in a proof assistant the terms `flip_AB_99` and `flip_AB_2099` will, by convention, stand for the proto-iso and for the iso respectively.

Now imagine a diagram editor working in tandem with a text editor that not only edits a formalization of the diagram, but also controls a proof assistant. It shouldn't be hard to make the occurrences of the term `flip_AB_99` be highlighted when the arrow '$\leftrightarrow_{99}$' is selected, and vice-versa. Allowing 2D diagrammatic "names" in the editor, as, e.g., the squares of Section 9, as synonyms for some ascii terms, can be added later as a separate feature.

22. A Database of Categorical Theorems

We, the people who think mostly diagrammatically, would like to have a database of theorems from Category Theory, all presented in a diagrammatical form.

This database would include the theorems I know, the theorems I have half-forgotten, and lots of theorems that I never knew, that were added to the database by other people.

That database already exists, but in a non-ideal, not-very-diagrammatical form: it is the published literature on Category Theory. Each paper has its own notion of "obviousness": when an author skips over details and claims that something is obvious, he supposes — in the spirit of section 10 — that the reader will be able to fill up the gaps.

I started to work on this subject because my notion of "obvious" seemed to be too different from the ones that I could find in the literature. The "archetypal models" being generalized are almost always mentioned in the papers,

but only in passing, and after all the axioms having been laid down; all the formal arguments are made in the abstract notation only; and informal notations can only be presented when they are used to prove new nontrivial results, and when their formalization has been completed — but then they are not "informal" anymore, and they have been promoted to "internal languages", "term calculi", "proof nets", "circuit diagrams", etc ([JS91] is an early example of an informal notation made formal, with a nice discussion).

As each notion of "obvious" is backed by a method for proof search, it ought to be possible to formalize — even if roughly — how each such proof-search method would work; and each different notion of what is "obvious" leads to a different way to present, and store, theorems. The same theorem T is remembered by a reader R_1 as T_1 and by a reader R_2 as T_2; as the two readers reconstruct the missing details the theorems grow to T_1', T_2', T_1'', T_2''. If we could put these objects side to side we could clarify how these different kinds of reasonings work.

It is possible to reason coherently with infinitesimals. This became clear when Non-Standard Analysis was invented, and then even clearer when Synthetical Differential Geometry came up. One way to prove that a way of reasoning is coherent is to model it mathematically — and to show that the model has good properties. NSA and SDG "validate" reasonings with infinitesimals. Similarly, one way to validate reasonings based on internal diagrams, archetypal languages and informal notations is to formalize how these reasonings work — and to build bridges between them and the standard ways.

Diagrams like the ones in sections 14 and 15 can be interpreted formally: a diagram induces a dictionary, which induces trees, which then let us interpret each object and arrow from the diagram as a λ-term. A full set of rules for interpreting diagrams can be developed — including ways to deal with lists of exceptions and hints — and so it would be conceivable to have a database of categorical definitions and theorems in which the entries would appear as diagrams (plus their associated hints), but at the same time they would have precise meanings, understood by a proof assistant...

Note that diagrams are becoming more and more meaningful mathematically in the last decades. Compare these ideas with the discussion in [Krö07], p.83, where he quotes this from [ES52], p.xi:

> 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.

23. Epilogue

This is an atypical paper. It has no theorems, and it doesn't prove any new mathematical truths — instead, we have shown several ways to represent constructions, and to structure proofs in several layers.

"Theorems" usually involve *equalities* between *constructions*, and hence they do not belong do the lowest layers; there are even ways — see Section 19 — to "project out" the parts of a theorem that involve equalities, and keep only the constructions... The result is a sort of a "syntactical skeleton" of the theorem.

Our "structuring" puts a situation with total information at the top, and below that, in different, parallel "legs", we put different partial views. Theorems do not belong to the *structuring*, either.

At some point we will have "meta-theorems" about properties of this structuring. But finding these meta-theorems should not be top priority right now: these meta-theorems will use known theorems, and at this point it is more important to put more known theorems in this format.

The above may look too vague, too philosophical (in contraposition to "mathematical"), too abstract. Category Theory used to be called "abstract nonsense"; the ideas of this paper, then, especially the more technical sections, 13 to 20, could look like "meta-abstract nonsense"... However, we are not pointing only towards the more abstract: we are showing ways — and reasons — to do the *general* in parallel with the *particular*, and to arrange the results of doing mathematics like that in forms that are better for *reconstruction* and *transmission*.

Our structuring allows the controlled use of *informal notations*. Several different informal notations can be used in parallel at the same time. This can be useful for comparing different people's notations, and for changing details on a notation and letting it evolve over time. The database of categorical knowledge proposed in the last section does not need a unified notation, and one of its functions could be to serve as a dictionary between notations — and as a way to make parts of the existing literature more accessible.

References

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

[Bel88] J. L. Bell. *Toposes and Local Set Theories*. Number 14 in Oxford Logic Guides. Oxford University Press, 1988.

[Bel08] J. L. Bell. *A Primer of Infinitesimal Analysis*. Cambridge, 2008.

[BJP10] J. P. Bernardy, P. Jansson, and R. Paterson. Parametricity and dependent types. *International Conference on Functional Programming*, 2010. http://www.cse.chalmers.se/~bernardy/ParDep/pardep.pdf.

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

[Fre76] P. Freyd. Properties invariant within equivalence types of categories. In A. Heller and M. Tierney, editors, *Algebra, Topology and Category Theory: A Collection of Papers in Honour of Samuel Eilenberg*, pages 55–61. Academic Press, 1976.

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

[Geu93] H. Geuvers. *Logics and Type Systems*. PhD thesis, University of Nijmegen, 1993.

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

[JS91] A. Joyal and R. Street. The geometry of tensor calculus i. *Advances in Mathematics*, 88:55–112, 1991.

[Koc77] A. Kock. A simple axiomatics for differentiation. *Mathematica Scandinavica*, 40(2):183–193, 1977. http://www.mscand.dk/article.php?id=2356.

[Koc80] A. Kock. *Synthetic Differential Geometry*. Cambridge, 1980. http://home.imf.au.dk/kock/sdg99.pdf.

[Krö07] R. Krömer. *Tool and Object: A History and Philosophy of Category Theory*. Birkhäuser, 2007.

[Law69] W. Lawvere. Adjointness in foundations. *Dialectica*, 23:281–296, 1969.

[Law70] W. Lawvere. Equality in hyperdoctrines and comprehension schema as an adjoint functor. *Proceedings of the American Mathematical Society Symposium on Pure Mathematics XVII*, 999:1–14, 1970.

[LS86] J. Lambek and P. Scott. *Introduction to Higher-Order Categorical Logic*. Cambridge, 1986.

[LS97] W. Lawvere and S. Schanuel. *Conceptual Mathematics: A first introduction to categories*. Cambridge, 1997.

[MR91] I. Moerdijk and G. E. Reyes. *Models for Smooth Infinitesimal Analysis*. Springer, 1991.

[Och10] E. Ochs. Downcasing types. Slides for a presentation at the UniLog'2010. http://angg.twu.net/math-b.html#unilog-2010, 2010.

[See83] R. A. G. Seely. Hyperdoctrines, natural deduction, and the beck condition. *Zeitschrift f. math. Logik und Grundlagen d. Math.*, 29:505–542, 1983.

[Tay86] P. Taylor. *Recursive Domains, Indexed Category Theory and Polymorphism*. PhD thesis, Cambridge, 1986.

[Wad89] P. Wadler. Theorems for free! In *Proc. FPCA'89*, pages 347–359. ACM, 1989.

Eduardo Ochs
LLaRC – Laboratório de Lógica e Representação do Conhecimento
Depto. de Física e Matemática, Pólo Universitário de Rio das Ostras
Universidade Federal Fluminense, Rio das ostras, RJ, Brazil
e-mail: eduardoochs@gmail.com

Submitted: November 1, 2010.
Revised: April 27, 2013.