Extra sections for the
"Internal Diagrams in Category Theory" paper —
part of a fuller set of changes, corrections, explanations —
first version, 2013apr05
Eduardo Ochs, eduardoochs@gmail.com

This: http://angg.twu.net/LATEX/2013idct-changes.pdf
the paper: http://angg.twu.net/LATEX/2013idct-changes.pdf

11.50. 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 the section [18.25. The syntactical world] 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 [18.75 The problem with toposes].

18.25. 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 [11.50. Skeletons of proofs].

We have been avoiding all mentions to equations between morphisms — for example, in the section [9. Adjunctions] 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 the course notes [http://angg.twu.net/math-b.html#unilog-2010].

18.50. Formalizing diagrams in Type Theory

We saw at sections [3. Downcased Types] and [7. Functors] 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 the diagram at p.22, where both $\delta^*\overline{\pi}^*(\delta^*Q)$ and δ^*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 meanings 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.

18.75 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. 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, 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.

[Todo: explain that the classifier object in a topos induces a hyperdoctrine structure on the underlying CCC, and it is that structure which is needed for interpreting first-order logic with equality there; the details are not as well-known as they should, and to add polymorphism we need to add extra structure which is even less familiar. I believe that these and other results in categorical semantics can become much more accessible with the use of diagrams in a downcased language, as in the sections [12. Hyperdoctrines] and [17. Objects of line type]; the diagrams for these semantics, translations, and their computer implementations are high-priority future work.]