Downcasing Types

(Abstract sent to UNILOG'2010)
Eduardo Ochs - LLaRC, PURO/UFF, Brazil
eduardoochs@gmail.com

For more details see:
http://angg.twu.net/LATEX/2009unilog-dnc.pdf

When we represent a category $\mathbf{C}$ in Type Theory it becomes a 7-uple: $(\mathbf{C}_0, \mathrm{Hom}_\mathbf{C}, \mathrm{id}_\mathbf{C}, \circ_\mathbf{C}; \mathsf{assoc}_\mathbf{C}, \mathsf{idL}_\mathbf{C}, \mathsf{idR}_\mathbf{C})$, where the first four components are "structure" and the last three are "properties".

We call the "structure" components the "syntactical part", and the "properties" components the "logical part". A *protocategory* is a 4-uple $(\mathbf{C}_0, \mathrm{Hom}_\mathbf{C}, \mathrm{id}_\mathbf{C}, \circ_\mathbf{C})$ — just the "syntactical skeleton" of what a category is, without the components that talk about equality of morphisms. By splitting at the right places the uples that represent functors, natural transformations, isos, adjunctions, limits, etc, we define proto-functors, proto-NTs, and so on.

The operation that takes entities and returns the corresponding proto-entities behaves as a projection, and we say that it goes from the "real world" — where everything has both a "syntactical" and a "logical" part — to the "syntactical world", where only the syntactical parts have been kept.

The opposite of to *project* is to *lift*. We may start with a proto-something, s^-, in the syntactical world, and try to lift it to an s in the real world that projects into s^-. Meta-theorems about lifting are hard to obtain, but we know many interesting liftings — each object r of the (projectable fragment of the) real world projects to an proto-object r^- that can be lifted back to r — and we can start by studying them to understand how liftings behave.

Proto-objects — even proto-proofs — are especially amenable to being represented diagrammatically, and there is a simple way to attribute a precise meaning to each entity — each node, arrow, etc — appearing in these diagrams. We will show how to formalize two such diagrammatic proofs — the Yoneda Lemma and one of the weakest monadicity theorems — as terms in Coq.

For most applications in Categorical Semantics one further trick is needed: "downcasing types", that lets us name entities by what they represent in the "archetypical case". For example, in a hyperdoctrine, if P is an object over $B \times C$ and $f : A \to B$ then Beck-Chevalley Condition for '$\forall$' says that the natural morphism from $f^*\Pi_{\pi_{BC}}P$ to $\Pi_{\pi_{AC}}(f \times C)^*P$ should be an iso. In the archetypical hyperdoctrine, Sub(**Set**), P "is" a subset $\{\, (b,c) \in B \times C \mid P(b,c) \,\}$ of $B \times C$, and both $f^*\Pi_{\pi_{BC}}P$ and $\Pi_{\pi_{AC}}(f \times C)^*P$ "deserve the name" $\{\, a \in A \mid \forall c \in C.\, P(fa, c) \,\}$. The downcasing of P is $b, c||P$, and the BCC map becomes a map $a||\forall c.P \mapsto a||\forall c.P$ that is not the identity, whose construction can be read out from a diagram.

Roughly, what is the happening is the following: the formal definition of hyperdoctrine generalizes *some* of the structure of Sub(**Set**); with our way of interpreting diagrams we can define all this structure diagrammatically, in a

notation that "suggests" that we are in Sub(**Set**), i.e., "in the archetypical case", and then we can "lift" these definitions to diagrams with the same two-dimensional structure, but in any of the standard notations.

Several categorical theorems become quite clear when we find "archetypical diagrams" for their (proto-)proofs, and then we lift those to standard notations; we will show some examples from Lawvere's "Adjointness in Foundations" (1969) and "Equality in Hyperdoctrines" (1970) papers.