A TYPE THEORY FOR FIBRED FUNCTORS AND POLYNOMIALS

MICHAEL SHULMAN

1. A dependent type theory with bunches

We have two kinds of types, which we call *types* and *functors*. Every type is a functor (a "constant functor", although the rules will not impose this), but not every functor is a type. We will refer to functors that are not known to be types as *nonconstant*.

As in Bunched Implication, there are two context-combining operations: an "additive" monoid structure (written $(\Delta\,;\Psi)$ with unit $\{\}_{\mathsf{a}}$) representing dependency, including the trivial case of cartesian products; and a "multiplicative" monoid structure (written $(\Delta\,,\Psi)$ with unit $\{\}_{\mathsf{m}}$) representing functor composition.

Because we allow dependency, we need these context operations to "happen in any context", so we have separate "telescopes" or "dependent contexts". The additive combination of contexts is potentially dependent, and hence ordered: in $(\Delta\,;\Psi)$ we could have Ψ dependent on Δ. But if it is not dependent, an exchange rule should be admissible, as are appropriate weakening, contraction, and substitution/cut. The multiplicative combination of contexts is not dependent: in $(\Delta\,,\Psi)$ we cannot have Ψ depend on Δ, though both can depend additively on the same context to the left. The multiplicative combination has no structural rules (unless you consider associativity and unitality to be a structural rule) — not even exchange, since functor composition is not commutative.

Finally, we also distinguish judgmentally those contexts and telescopes that consist entirely of types combined additively, which play a special role. (They are the contexts of the "underlying ordinary type theory" that we are enhancing with functors.) The judgments are shown in Figure 1, along with some probably admissible rules saying that every type is a functor and similarly for contexts in Figure 2. The rules for context combinations are shown in Figure 3 and the judgmental equality rules making these into monoids in Figure 4. In Figure 5 we give a "variable" rule for additive contexts only; the natural multiplicative variable rule "$x : A \vdash x : A$" (with no ambient context) is a special case of this, since unary multiplicative contexts are the same as unary additive ones.

Date: April 2018.

$$\Gamma\ \mathsf{ctx} \qquad \Gamma\ \mathsf{typectx} \qquad \Gamma \vdash A\ \mathsf{type} \qquad \Gamma \vdash F\ \mathsf{functor}$$

$$\Gamma \vdash \Delta\ \mathsf{tel} \qquad \Gamma \vdash \Delta\ \mathsf{typetel} \qquad \Gamma \vdash a : A$$

Figure 1. Judgments

$$\frac{\Gamma\ \mathsf{typectx}}{\Gamma\ \mathsf{ctx}} \qquad \frac{\Gamma \vdash A\ \mathsf{type}}{\Gamma \vdash A\ \mathsf{functor}} \qquad \frac{\Gamma \vdash \Delta\ \mathsf{typetel}}{\Gamma \vdash \Delta\ \mathsf{tel}}$$

FIGURE 2. Inclusion judgments (probably admissible)

$$\frac{}{\diamond\ \mathsf{ctx}} \qquad \frac{\Gamma\ \mathsf{ctx} \qquad \Gamma \vdash \Delta\ \mathsf{tel}}{(\Gamma\,;\Delta)\ \mathsf{ctx}} \qquad \frac{\Gamma\ \mathsf{typectx} \qquad \Gamma \vdash \Delta\ \mathsf{typetel}}{(\Gamma\,;\Delta)\ \mathsf{typectx}} \qquad \frac{\Gamma\ \mathsf{typectx}}{\Gamma \vdash \{\}_{\mathsf{m}}\ \mathsf{tel}}$$

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \{\}_{\mathsf{a}}\ \mathsf{tel}} \qquad \frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \{\}_{\mathsf{a}}\ \mathsf{typetel}} \qquad \frac{\Gamma \vdash A\ \mathsf{functor}}{\Gamma \vdash (x : A)\ \mathsf{tel}} \qquad \frac{\Gamma \vdash A\ \mathsf{type}}{\Gamma \vdash (x : A)\ \mathsf{typetel}}$$

$$\frac{\Gamma \vdash \Delta\ \mathsf{tel} \qquad \Gamma\,;\Delta \vdash \Psi\ \mathsf{tel}}{\Gamma \vdash (\Delta\,;\Psi)\ \mathsf{tel}} \qquad \frac{\Gamma \vdash \Delta\ \mathsf{typetel} \qquad \Gamma\,;\Delta \vdash \Psi\ \mathsf{typetel}}{\Gamma \vdash (\Delta\,;\Psi)\ \mathsf{typetel}}$$

$$\frac{\Gamma\ \mathsf{typectx} \qquad \Gamma \vdash \Delta\ \mathsf{tel} \qquad \Gamma \vdash \Phi\ \mathsf{tel} \qquad \Gamma\,;\Delta \vdash \Psi\ \mathsf{tel}}{\Gamma\,;\Delta \vdash (\Phi\,,\Psi)\ \mathsf{tel}}$$

FIGURE 3. Rules for telescopes and contexts

$$\frac{\Gamma\ \mathsf{ctx}}{(\Gamma\,;\{\}_{\mathsf{a}}) \equiv \Gamma\ \mathsf{ctx}} \qquad \frac{\Gamma \vdash \Delta\ \mathsf{tel} \qquad \Gamma\,;\Delta \vdash \Psi\ \mathsf{tel}}{(\Gamma\,;(\Delta\,;\Psi)) \equiv ((\Gamma\,;\Delta)\,;\Psi)\ \mathsf{ctx}} \qquad \frac{\Gamma \vdash \Delta\ \mathsf{tel}}{\Gamma \vdash (\{\}_{\mathsf{a}}\,;\Delta) \equiv \Delta\ \mathsf{tel}}$$

$$\frac{\Gamma \vdash \Delta\ \mathsf{tel}}{\Gamma \vdash (\Delta\,;\{\}_{\mathsf{a}}) \equiv \Delta\ \mathsf{tel}} \qquad \frac{\Gamma \vdash \Delta\ \mathsf{tel} \qquad \Gamma\,;\Delta \vdash \Psi\ \mathsf{tel} \qquad \Gamma\,;\Delta\,;\Psi \vdash \Phi\ \mathsf{tel}}{\Gamma \vdash (\Delta\,;(\Psi\,;\Phi)) \equiv ((\Delta\,;\Psi)\,;\Phi)\ \mathsf{tel}}$$

$$\frac{\Gamma\ \mathsf{typectx} \qquad \Gamma \vdash \Delta\ \mathsf{tel} \qquad \Gamma\,;\Delta \vdash \Phi\ \mathsf{tel}}{\Gamma\,;\Delta \vdash (\{\}_{\mathsf{m}}\,,\Delta) \equiv \Delta\ \mathsf{tel}} \qquad \frac{\Gamma\ \mathsf{typectx} \qquad \Gamma \vdash \Phi\ \mathsf{tel}}{\Gamma \vdash (\Delta\,,\{\}_{\mathsf{m}}) \equiv \Phi\ \mathsf{tel}}$$

$$\frac{\Gamma\ \mathsf{typectx} \qquad \Gamma \vdash \Delta\ \mathsf{tel} \qquad \Gamma \vdash \Phi\ \mathsf{tel} \qquad \Gamma \vdash \Psi\ \mathsf{tel} \qquad \Gamma\,;\Delta \vdash \Upsilon\ \mathsf{tel}}{\Gamma\,;\Delta \vdash (\Phi\,,(\Psi\,,\Upsilon)) \equiv ((\Phi\,,\Psi)\,,\Upsilon)\ \mathsf{tel}}$$

plus congruence rules

FIGURE 4. Equality rules for telescopes and contexts

$$\frac{\Gamma \vdash A\ \mathsf{functor} \qquad \Gamma, x : A \vdash \Delta\ \mathsf{tel}}{\Gamma\,; x : A\,;\Delta \vdash x : A}$$

FIGURE 5. Variable rule for additive contexts

Unlike in (the original presentation of) ordinary BI, we make the structural rules for additive contexts admissible. We also allow multiplicative combination to happen in an extra additive context, as long as the nonconstant functors in that context

$$\frac{\Gamma \text{ typectx} \qquad \Gamma \vdash \Delta \text{ tel} \qquad \Gamma \vdash F \text{ functor} \qquad \Gamma\,;\Delta \vdash G \text{ functor}}{\Gamma\,;\Delta \vdash F \circ G \text{ functor}} \qquad \frac{\Gamma \text{ typectx}}{\Gamma \vdash \mathbb{X} \text{ functor}}$$

$$\frac{\Gamma \text{ typectx} \qquad \Gamma \vdash \Delta \text{ tel} \qquad \Gamma \vdash F \text{ functor} \qquad \Gamma\,;\Delta \vdash G \text{ functor}}{\Gamma\,;\Delta\,;(\phi : F\,,\psi : G) \vdash \langle \phi, \psi \rangle : F \circ G} \qquad \frac{\Gamma \text{ typectx}}{\Gamma\,;\{\}_{\mathsf{m}} \vdash \maltese : \mathbb{X}}$$

$$\frac{\Gamma(z : \mathbb{X}) \vdash H \text{ functor} \qquad \Gamma(\{\}_{\mathsf{m}})[\maltese/z] \vdash \theta : H[\maltese/z]}{\Gamma(z : \mathbb{X}) \vdash (\mathsf{let}\, \maltese \coloneqq z\, \mathsf{in}\, \theta) : H}$$

$$\frac{\Gamma(z : F \circ G) \vdash H \text{ functor} \qquad \Gamma(x : F, y : G)[\langle x, y \rangle/z] \vdash \theta : H[\langle x, y \rangle/z]}{\Gamma(z : F \circ G) \vdash (\mathsf{let}\, \langle x, y \rangle \coloneqq z\, \mathsf{in}\, \theta) : H}$$

plus computation rules

FIGURE 6. Rules for multiplicative tensor

are only used on the right-most telescope in the multiplicative combination. This appears in the last rule in Figure 3. That is, the multiplicative combination satisfies a sort of "ordered modal context-clearing" with respect to nonconstant functors: none of them can be used on the left-hand branch.

An analogous restriction appears in the formation and introduction rules for the multiplicative connectives in Figure 6, which internalize functor composition and identities. For simplicity, we give the rules in sequent calculus style rather than building in cuts. We write the binary multiplicative tensor as $\circ$, thinking of it as functor composition, and the unary one as $\mathbb{X}$, thinking of it as the identity functor $X \mapsto X$.

As in ordinary BI, in the elimination rules for $\mathbb{X}$ and $\circ$ we have a context Γ with a "hole" that gets filled by different things; this is what notation like $\Gamma(z : \mathbb{X})$ means. However, dependency makes things a little complicated. When defining the body of the let-expression and when typing the result, we have to substitute in the rest of the context that depends on the hole variable as well as in the consequent. (This is like the "Frobenius" version of the eliminator for identity types; as in that case, it could probably be eliminated if we had enough function types, but our intended model does not.) If we built in cuts, then in describing the term being let-destructed we would have to take account of the part of the context on which it can depend.

The rules for negative additive Σs are shown in Figure 7, again in sequent calculus style. They basically internalize the additive context combination, except that we also allow destructing Σs on the left of a multiplicative context join. Note that $\mathbf{1}$ is a type, and $\sum_{x:A} B$ is a type if A and B are.

The rules for additive Πs are shown in Figure 8. We only assert that products of functors over types exist, while products of types over types are types. Otherwise there is nothing surprising here. As usual, if F does not depend on A, we write $A \to F$ instead of $\prod_{x:A} F$.

$$\frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \mathbf{1}\ \mathsf{type}} \qquad \frac{\Gamma \vdash F\ \mathsf{functor} \qquad \Gamma\,;x : F \vdash G\ \mathsf{functor}}{\Gamma \vdash \sum_{x:F} G\ \mathsf{functor}}$$

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma\,;x : A \vdash B\ \mathsf{type}}{\Gamma \vdash \sum_{x:A} B\ \mathsf{type}} \qquad \frac{\Gamma\ \mathsf{ctx}}{\Gamma \vdash \star : \mathbf{1}} \qquad \frac{\Gamma \vdash \phi : F \qquad \Gamma \vdash \psi : G[\phi/x]}{\Gamma \vdash (\phi, \psi) : \sum_{x:F} G}$$

$$\frac{\Gamma \vdash a : A}{\Gamma\,;(u : \mathbf{1}\,, \Delta) \vdash (\mathsf{let}\, \star \coloneqq u\, \mathsf{in}\, a) : A}$$

$$\frac{\Gamma, (z : \sum_{x:A} B\,, \Delta) \vdash C\ \mathsf{functor} \qquad \Gamma\,;x : A\,;(y : B\,, \Delta) \vdash c : C[(x,y)/z]}{\Gamma\,;(z : \sum_{x:A} B\,, \Delta) \vdash (\mathsf{let}\,(x,y) \coloneqq z\, \mathsf{in}\, c) : C}$$

plus computation and uniqueness rules

FIGURE 7. Rules for additive Σs

$$\frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma\,;x : A \vdash F\ \mathsf{functor}}{\Gamma \vdash \prod_{x:A} F\ \mathsf{functor}} \qquad \frac{\Gamma \vdash A\ \mathsf{type} \qquad \Gamma\,;x : A \vdash B\ \mathsf{type}}{\Gamma \vdash \prod_{x:A} B\ \mathsf{type}}$$

$$\frac{\Gamma\,;x : A \vdash \phi : F}{\Gamma \vdash \lambda x.\phi : \prod_{x:A} F} \qquad \frac{\Gamma \vdash f : \prod_{x:A} F \qquad \Gamma \vdash a : A}{\Gamma \vdash f(a) : F[a/x]}$$

plus computation and uniqueness rules

FIGURE 8. Rules for additive Πs

2. Polynomial functors and the Yoneda lemma

The original goal of all this was to be able to use the "functors" and maps between them to describe the "attaching maps" for constructors of higher inductive types. The domains of such maps are *polynomial functors*, which in our theory we can define in any type context:

$$\cfrac{\cfrac{\cfrac{\cfrac{\Gamma\ \mathsf{typectx} \qquad \Gamma \vdash A\ \mathsf{type} \qquad \Gamma\,;x : A \vdash B_x\ \mathsf{type}}{(\Gamma\,;x : A\,;y : B_x)\ \mathsf{typectx}}}{\Gamma\,;x : A\,;y : B_x \vdash \mathbb{X}\ \mathsf{functor}}}{\Gamma\,;x : A \vdash (B_x \to \mathbb{X})\ \mathsf{functor}}}{\Gamma \vdash \sum_{x:A} (B_x \to \mathbb{X})\ \mathsf{functor}}$$

The codomains of the attaching maps will be functors obtained (by further yet-to-be-specified rules) from the "monad associated to the previous constructors". Thus, we need to be able to define maps out of polynomial functors.

Internally, a polynomial functor can be thought of as a "coproduct of representable functors". Therefore, by the Yoneda lemma, maps from $\sum_{x:A}(B_x \to \mathbb{X})$ to another functor F should be determined by a family of elements of $F(B_x)$ for each $x : A$. We don't have a notion of "application" of functors to types; instead we just compose a functor with a type (considered as a constant functor). With this representation,

using the rules for $\circ$ we can derive maps out of polynomial functors from such "families of elements". First we have (omitting some variables and terms):

$$\dfrac{\dfrac{\dfrac{\Gamma\,;x:A\vdash F\circ B_x}{\Gamma\,;x:A\,;(B_x\to\mathbb{X})\vdash F\circ B_x}\,\textsc{weak}\qquad\dfrac{??}{\Gamma\,;x:A\,;(B_x\to\mathbb{X})\,;F\circ B_x\vdash F}}{\Gamma\,;x:A\,;(B_x\to\mathbb{X})\vdash F}\,\textsc{cut}}{\Gamma\,;\sum_{x:A}(B_x\to\mathbb{X})\vdash F}\,\Sigma\textsc{-left}$$

Here the left-hand premise $\Gamma\,;x:A\vdash F\circ B_x$ is "a family of elements of $F(B_x)$ for each $x:A$" while the conclusion is the map we want. Thus it remains to derive the right-hand premise.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\Gamma\,;x:A\,;F\vdash F}\,\textsc{var}\qquad\dfrac{}{\Gamma\,;x:A\,;(B_x\to\mathbb{X})\,;B_x\vdash\mathbb{X}}\,\Pi\textsc{-app}}{\Gamma\,;x:A\,;(B_x\to\mathbb{X})\,;(F\,,B_x)\vdash F\circ\mathbb{X}}\,\circ\textsc{-intro}}{\Gamma\,;x:A\,;(B_x\to\mathbb{X})\,;F\circ B_x\vdash F\circ\mathbb{X}}\,\circ\textsc{-elim}}{\Gamma\,;x:A\,;(B_x\to\mathbb{X})\,;F\circ B_x\vdash F}\,\textsc{unit}$$

The final step is the isomorphism $F\circ\mathbb{X}\cong F$, which is proved as usual. Note how the use of $\circ$-introduction obeys the restriction that only the right-hand functor can depend on nonconstant functors in the context.

The setup of section 1, allowing us to nest additive and multiplicative contexts arbitrarily deeply, may seem like kind of overkill for this simple application. But it's actually the easiest way I was able to think of to say exactly what we are allowed to do with these contexts, in a way that's flexible enough to permit the above argument. We do need some nesting: the crucial step above involves the nested context $((B_x\to\mathbb{X})\,;(F\,,B_x))$. And we can't just separate the type context from the functor context either, since B_x is a type but has to appear in the functor context.

The setup does allow us to prove other useful things. For instance, since we allowed eliminating Σs on the left of a multiplicative context, we can show that they are "objectwise" in the sense that they act on the first functor in a composite. Given $x:A\vdash F_x$ $\mathsf{functor}$ and $\vdash G$ $\mathsf{functor}$, one direction is easy:

$$\dfrac{\dfrac{\dfrac{}{x:A\,;F_x\vdash\sum_{x:A}F_x}\qquad\dfrac{}{x:A\,;G\vdash G}}{x:A\,;(F_x\,,G)\vdash\sum_{x:A}F_x\circ G}}{\sum_{x:A}(F_x\circ G)\vdash\sum_{x:A}F_x\circ G}$$

and the other is somewhat harder:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x:A\,;(y:F_x\,,z:G)\vdash x:A}}{y:\sum_{x:A}F_x\,,z:G\vdash\pi_1(y):A}}{\sum_{x:A}F_x\circ G\vdash a:A}\qquad\dfrac{\dfrac{y:\sum_{x:A}F_x\vdash\pi_2(y):F_a\qquad G\vdash G}{\sum_{x:A}F_x\,,G\vdash F_a\circ G}}{\sum_{x:A}F_x\circ G\vdash F_a\circ G}}{\sum_{x:A}F_x\circ G\vdash\sum_{x:A}(F_x\circ G)}$$

It should be straightforward to show that these are inverses. We expect Πs to also be objectwise, and in one direction we have:

$$\dfrac{\dfrac{\dfrac{\overline{x : A \,;\, \prod_{x:A} F_x \vdash F_x} \qquad \overline{G \vdash G}}{x : A \,;\, (\prod_{x:A} F_x \,,\, G) \vdash F_x \circ G}}{\prod_{x:A} F_x \circ G \,;\, x : A \vdash F_x \circ G}}{\prod_{x:A} F_x \circ G \vdash \prod_{x:A} (F_x \circ G)}$$

but I have not been able to prove the opposite implication, or think of any enhancement of the rules in Figure 8 that would make it possible. It may be necessary to assert axiomatically that the above definable unidirectional map is an isomorphism; I guess this is not hugely surprising given that other axioms like function extensionality are necessary when working with Πs.

We also have the "non-axiom of choice" for functors. Suppose we are given

$$\begin{aligned} \Gamma \ \mathsf{typectx} \\ \Gamma \,;\, x : A \vdash B_x \ \mathsf{type} \\ \Gamma \,;\, x : A \,;\, y : B_x \vdash F_{x,y} \ \mathsf{functor} \end{aligned}$$

Then we can form both

$$\begin{aligned} \Gamma \vdash \textstyle\prod_{x:A} \sum_{y:B_x} F_{x,y} \ \mathsf{functor} \\ \Gamma \vdash \textstyle\sum_{g:\prod_{x:A} B_x} \prod_{x:A} F_{x,g(x)} \ \mathsf{functor} \end{aligned}$$

and we can derive inverse isomorphisms between them using the usual formulas.

It follows that polynomial functors are closed under Σs and Πs over types. Combining all the above properties (including the axiomatic objectwise-ness for Π), we can also show that they are closed under composition:

$$\begin{aligned} (\textstyle\sum_{x:A}(B_x \to \mathbb{X})) \circ (\sum_{u:C}(D_u \to \mathbb{X})) &\cong \textstyle\sum_{x:A} \prod_{y:B_x} (\mathbb{X} \circ (\sum_{u:C} \prod_{v:D_u} \mathbb{X})) \\ &\cong \textstyle\sum_{x:A} \prod_{y:B_x} \sum_{u:C} \prod_{v:D_u} \mathbb{X} \\ &\cong \textstyle\sum_{x:A} \sum_{g:B_x \to C} \prod_{y:B_x} \prod_{v:D_{g(y)}} \mathbb{X} \\ &\cong \textstyle\sum_{(x,g):\sum_{x:A}(B_x \to C)} \prod_{w:B_x \times D_{g(y)}} \mathbb{X} \end{aligned}$$

It is therefore sensible to introduce a judgment to characterize polynomial functors inductively:

$$\dfrac{\Gamma \ \mathsf{typectx}}{\Gamma \vdash \mathbb{X} \ \mathsf{poly}} \qquad \dfrac{\Gamma \vdash F \ \mathsf{poly} \qquad \Gamma \vdash G \ \mathsf{poly}}{\Gamma \vdash F \circ G \ \mathsf{poly}}$$

$$\dfrac{\Gamma \vdash A \ \mathsf{type} \qquad \Gamma, x : A \vdash F \ \mathsf{poly}}{\Gamma \vdash \textstyle\sum_{x:A} F \ \mathsf{poly}} \qquad \dfrac{\Gamma \vdash A \ \mathsf{type} \qquad \Gamma, x : A \vdash F \ \mathsf{poly}}{\Gamma \vdash \textstyle\prod_{x:A} F \ \mathsf{poly}}$$

I think this will be useful when describing HITs. (In fact, so far we have not introduced any rules that produce *non*-polynomial functors, but the original idea was to introduce free monads on them.)

3. A sketch of semantics

Let $\mathcal{C}$ be a category with finite limits. A **fibred (endo)functor** of $\mathcal{C}$ is a functor $\mathcal{C}^{\to} \to \mathcal{C}^{\to}$ that fixes codomains and preserves pullback squares. The category of fibred functors has finite limits taken pointwise. A morphism of fibred functors

(i.e. a fibred natural transformation) is **cartesian** if its naturality squares are pullbacks. Every object A of $\mathcal{C}$ yields a constant fibred functor taking every $X \to \Xi$ to $\Xi \times A \to \Xi$. The functors arising in this way are precisely those whose unique map to the terminal functor is cartesian.

We interpret the contexts, telescopes, and functors of our type theory by fibred functors and morphisms between them in the usual way. We interpret types and type telescopes by cartesian morphisms, so that type contexts are interpreted by objects of $\mathcal{C}$. The additive context combination, and the Σs that internalize it, are just the usual composition of dependent projections.

The multiplicative context combination, and the multiplicative product and unit that internalize it, are defined using functor composition, but using the fibred nature of our functors. The multiplicative unit ($\{\}_{\mathsf{m}}$ or $\mathbb{X}$), in type context $\Gamma \in \mathcal{C}$, is interpreted by the fibred functor that sends every $X \to \Xi$ to the projection $X \times \Gamma \to X \to \Xi$. This factors alternatively through $\Xi \times \Gamma$, i.e. the constant functor at Γ, giving the dependency $\Gamma \vdash \mathbb{X}\ \mathsf{functor}$.

Composition is somewhat trickier. Given a type context $\Gamma \in \mathcal{C}$ and a fibred functor Δ mapping to Γ, and also fibred functors F over Γ and G over Δ, we interpret $(F\,, G)$ and $F \circ G$ by the functor defined as follows. Given $X \to \Xi$, we first apply G to get

$$G_\Xi(X) \to \Delta_\Xi(X) \to \Xi \times \Gamma \to \Xi.$$

Then we apply F to $G_\Xi(X)$ regarded as indexed, not over Ξ, but over $\Delta_\Xi(X)$. This gives

$$F_{\Delta_\Xi(X)}(G_\Xi(X)) \to \Delta_\Xi(X) \times \Gamma \to \Xi \times \Gamma \times \Gamma \to \Xi.$$

Finally we pull this back along the diagonal $\Gamma \to \Gamma \times \Gamma$ to get

$$(F \circ G)_\Xi(X) \to \Delta_\Xi(X) \to \Xi \times \Gamma \to \Xi.$$

Associativity and unitality are straightforward to check.

Finally, assuming $\mathcal{C}$ to be locally cartesian closed, we define Πs objectwise: given a cartesian map $A \to \Gamma$ and a map $F \to A$, we define $\prod_{x:A} F$ at X to be the dependent product of $F(X) \to A(X)$ along $A(X) \to \Gamma(X)$. The Beck-Chevalley condition for dependent products, and the cartesianness of $A \to \Gamma$, ensures that this is again a fibred functor. (This is why we allow only types, rather than arbitrary functors, in the domain of Πs.)

If $\mathcal{C}$ has a homotopy theory, then we can restrict the *types* to be fibrations. We do not restrict the functors in any way. In particular, functors need not preserve fibrations; but this is not a problem, because we have not syntactically required the "application" $F \circ A$ of a functor to a type to result in a type. In this model the constant functor at a non-fibration is a perfectly good functor, but not a type.

4. Codensity monads

In general, functor composition $\circ$ is not a *closed* monoidal structure. However, if A is a type regarded as a constant functor, then $(- \circ A)$ does have a right adjoint defined on constant functors, obtained by right Kan extension along the functor from the terminal category that picks out A. The result may no longer be fibrant, though, so we don't assert that it is a type.

$$\frac{\Gamma\ \mathsf{typectx} \qquad \Gamma \vdash A\ \mathsf{type} \qquad \Gamma \vdash B\ \mathsf{type}}{\Gamma \vdash A \multimap B\ \mathsf{functor}} \qquad\qquad \frac{\Gamma\,;(\Delta\,, x : A) \vdash b : B}{\Gamma\,;\Delta \vdash (\lambda\!\!\lambda x.b) : A \multimap B}$$