Polynomial Universes & Natural Models

C.B. Aberlé

November 22, 2024

Dependent types are ubiquitous in mathematics, pure and applied. When we say "let v be a vector of length n," we make the collection of values to which v may belong *dependent* upon the value of n. Such dependency of types-of-things on values-of-things is fundamental to our ability to express complex mathematical ideas and build up sophisticated abstractions. By taking this essential idea to heart, dependent type theory provides all of the following:

- An elegant **syntax** for expressing mathematical ideas.
- **Computational** interpretations that allow automation and formal verification of mathematical proofs.
- A robust **categorical semantics** that allow type theoretic syntax to be used in order to work in the internal languages of well-structured categories, and even more exotic structures, such as ∞-categories.

Of these, it is the **categorical semantics** of dependent type theory that shall be the focus of this blog post. Specifically, although these categorical semantics are evidently quite powerful, they are also notoriously subtle and difficult to get right, owing mainly to the issue of *strictness*. As we shall see, one often needs to *strictify* various identities that typically hold only up to isomorphism in arbitrary categories, but that must hold *strictly* in order for such a category to soundly model the type-theoretic syntax.

Perhaps surprisingly – or unsurprisingly to regular readers of this blog – a key device in resolving this difficulty turns out to be the categorical machinery of *polynomial functors*. Awodey – and later Newstead – have shown that there is a strong connection between dependent type theory and polynomial functors, via their concept of natural models, which cleanly solves the strictification problem for the categorical semantics of dependent type theory. In particular, the solution to this problem offered by Awodey and Newstead makes use of the type-theoretic concept of a *universe.* Such universes turn out to naturally be regarded as polynomial functors on a suitably-chosen category of presheaves.

In this post, I will summarize the work I conducted over the summer at Topos, together with David Spivak, wherein we have aimed to extend, refine, and generalize this treatment of models of type theory by studying properties of such *polynomial universes*. We shall see how the language

of polynomial functors can be used to elegantly capture all of the usual constructs of dependent type theory, and moreover, how interpreting this language in *homotopy type theory* reveals striking properties of these structures.

1 Recap of Dependent Type Theory & Natural Models

In Martin-Löf Type Theory (MLTT) we have the primitive judgments:

$$\Gamma\ \mathsf{Ctx} \qquad \Gamma \vdash A\ \mathsf{Type} \qquad \Gamma \vdash a : A \qquad \Gamma \vdash a_0 \equiv a_1 : A$$

which are then subject to the following rules:

$$\frac{}{\epsilon\ \mathsf{Ctx}} \qquad \frac{\Gamma\ \mathsf{Ctx} \qquad \Gamma \vdash A\ \mathsf{Type}}{\Gamma, x : A\ \mathsf{Ctx}}$$

$$\frac{}{\Gamma \vdash 1\ \mathsf{Type}} \qquad \frac{}{\Gamma \vdash () : 1} \qquad \frac{\Gamma \vdash u : 1}{\Gamma \vdash u = () : 1}$$

$$\frac{\Gamma \vdash A\ \mathsf{Type} \qquad \Gamma, x : A \vdash B\ \mathsf{Type}}{\Gamma \vdash \Sigma x : A.B\ \mathsf{Type}} \qquad \frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : B[a/x]}{\Gamma \vdash (a, b) : \Sigma x : A.B}$$

$$\frac{\Gamma \vdash p : \Sigma x : A.B}{\Gamma \vdash \pi_1(p) : A} \qquad \frac{\Gamma \vdash p : \Sigma x : A.B}{\Gamma \vdash \pi_2(p) : B[\pi_1(p)/x]}$$

$$\frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : B[a/x]}{\Gamma \vdash \pi_1(a, b) \equiv a : A} \qquad \frac{\Gamma \vdash a : A \qquad \Gamma \vdash b : B[a/x]}{\Gamma \vdash \pi_2(p) \equiv b : B[a/x]}$$

$$\frac{\Gamma \vdash p \in \Sigma x : A.B}{\Gamma \vdash p \equiv (\pi_1(p), \pi_2(p)) : \Sigma x : A.B}$$

$$\frac{\Gamma \vdash A\ \mathsf{Type} \qquad \Gamma, x : A \vdash B\ \mathsf{Type}}{\Gamma \vdash \Pi x : A.B\ \mathsf{Type}} \qquad \frac{\Gamma, x : A \vdash f : B}{\Gamma \vdash \lambda x.f : \Pi x : A.B}$$

$$\frac{\Gamma \vdash f : \Pi x : A.B \qquad \Gamma \vdash a : A}{\Gamma \vdash f(a) : B[a/x]} \qquad \frac{\Gamma \vdash f : \Pi x : A.B \qquad \gamma \vdash a : A}{\Gamma \vdash (\lambda x.f)(a) \equiv f[a/x] : B[a/x]}$$

$$\frac{\Gamma \vdash f \in \Pi x : A.B}{\Gamma \vdash f \equiv \lambda x.f(x) : \Pi x : A.B}$$

where the operation of *substitution* is defined as follows:

$$\begin{array}{rcl}
1[a/x] & = & 1 \\
()[a/x] & = & () \\
(\Sigma y : \mathrm{A}.\mathrm{B})[a/x] & = & \Sigma y : \mathrm{A}[a/x].\mathrm{B}[a/x] \\
(b, c)[a/x] & = & (b[a/x], c[a/x]) \\
\pi_1(p)[a/x] & = & \pi_1(p[a/x]) \\
\pi_2(p)[a/x] & = & \pi_2(p[a/x]) \\
(\Pi y : \mathrm{A}.\mathrm{B})[a/x] & = & \Pi y : \mathrm{A}[a/x].\mathrm{B}[a/x] \\
(\lambda y.f)[a/x] & = & \lambda y.f[a/x] \\
f(b)[a/x] & = & f[a/x](b[a/x])
\end{array}$$

Although contexts may seem rather trivial from a syntactic perspective, they are key to understanding the categorical semantics of dependent type theory. In particular, when modeling a dependent type theory as a category, it is the *contexts* which form the objects of this category, with morphisms between contexts being *substitutions* of terms in the domain context for the variables of the codomain context. A type A dependent upon variables in a context Γ is then interpreted as a morphism (i.e. substitution) $\Gamma, x : \mathrm{A} \to \Gamma$, called a *display map*, whose domain represents the context Γ extended with a variable of type A. We then interpret a term a of type A in context Γ as a *section* of the display map representing A, i.e.

$$\begin{array}{ccc}
\Gamma & \xrightarrow{a} & \Gamma, x : \mathrm{A} \\
& \searrow & \downarrow{\scriptstyle \mathrm{A}} \\
& & \Gamma
\end{array}$$

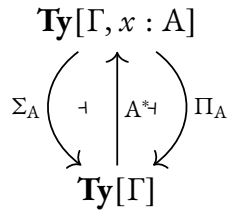
Hence for each context Γ, there is a category $\mathbf{Ty}[\Gamma]$, which is the full subcategory of the slice category $\mathcal{C}/\Gamma$ consisting of all display maps, wherein objects correspond to types in context Γ, and morphisms correspond to terms.

In typical categorical semantics, given a substitution $f : \Gamma \to \Delta$, and a display map $\mathrm{A} : \Delta, x : \mathrm{A} \to \Delta$, we then interpret the action of applying the substitution f to A as a pullback:

$$\begin{array}{ccc}
\Gamma, x : \mathrm{A}[f] & \longrightarrow & \Delta, x : \mathrm{A} \\
{\scriptstyle \mathrm{A}[f]}\downarrow & \lrcorner & \downarrow{\scriptstyle \mathrm{A}} \\
\Gamma & \xrightarrow[f]{} & \Delta
\end{array}$$

In particular, then, any display map $\mathrm{A} : \Gamma, x : \mathrm{A} \to \Gamma$ induces a functor $\mathbf{Ty}[\Gamma] \to \mathbf{Ty}[\Gamma, x : \mathrm{A}]$ by substitution along A, which corresponds to *weakening* a context by adding a variable of type A. The left and right adjoints to the weakening functor (if they exist) then correspond to Σ and

Π types, respectively.



In order for the operation of forming Σ and Π types to be stable under substitution (i.e. pullback), these must additionally satisfy the *Beck-Chevalley* condition:

$$\begin{array}{ccc} \Gamma, x : \mathrm{A}[f] & \xrightarrow{g} & \Delta, x : \mathrm{A} \\ {\scriptstyle \mathrm{A}[f]}\downarrow & \lrcorner & \downarrow{\scriptstyle \mathrm{A}} \\ \Gamma & \xrightarrow[f]{} & \Delta \end{array} \quad \Longrightarrow \quad (\Sigma_{\mathrm{A}}(\mathrm{B}))[f] \xleftarrow{\simeq} \Sigma_{\mathrm{A}[f]}(\mathrm{B}[g])$$

So far, we have told a pleasingly straightforward story of how to interpret the syntax of dependent type theory categorically. Unfortunately, this story is a fantasy, and the interpretation of type-theoretic syntax into categorical semantics sketched above is unsound, as it stands. The problem in essentials is that, in the syntax of type theory, substitution is *strictly* associative, and *strictly* satisfies the Beck-Chevalley condition. However, in the above categorical semantics, substitution by pullback, which is in general only associative up to isomorphism, and likewise for the Beck-Chevalley condition. It is precisely this problem which natural models exist to solve.

As mentioned previously, the key insight in formulating the notion of a natural model is that the problem of strictness in the semantics of type theory has, in a sense, already been solved by the notion of *type universes*. A universe is a type $\mathcal{U}$ whose elements can themselves be regarded as types, as follows:

$$\frac{}{\Gamma \vdash \mathcal{U}\ \mathsf{Type}} \qquad \frac{\Gamma \vdash \mathrm{A} : \mathcal{U}}{\Gamma \vdash \mathcal{U}_\bullet(\mathrm{A})\ \mathsf{Type}}$$

In categorical semantics, we interpret such a universe as a display map $u : \mathcal{U}_\bullet \to \mathcal{U}$. A type in context Γ may then be instead represented as a morphism $\mathrm{A} : \Gamma \to \mathcal{U}$, whereby we obtain the corresponding display map for A as the pullback:

$$\begin{array}{ccc} \Gamma, x : \mathrm{A} & \longrightarrow & \mathcal{U}_\bullet \\ \downarrow & \lrcorner & \downarrow{\scriptstyle u} \\ \Gamma & \xrightarrow[\mathrm{A}]{} & \mathcal{U} \end{array}$$

Conversely, we say that a display map $\Gamma, x : \mathrm{A} \to \Gamma$ is *classified* by $\mathcal{U}$ if there are morphisms forming a pullback square as above.

Hence given a universe of types $\mathcal{U}$, rather than representing substitution as pullback, we can simply represent the action of applying a substitution $f : \Gamma \to \Delta$ to a family of types $\mathrm{A} : \Delta \to \mathcal{U}$ as the precomposition $\mathrm{A} \circ f : \Gamma \to \mathcal{U}$, which is automatically strictly associative,

and strictly satisfies the Beck-Chevalley condition for Σ types/Π types if $\mathcal{U}$ is suitably closed under Σ types / Π types, respectively.

To interpret the syntax of dependent type theory in a category C of contexts and substitutions, it therefore suffices to *embed* C into a category whose type-theoretic internal language possesses such a universe whose types correspond to those of C. And what better embedding of a category C into a larger category with a well-understood type-theoretic language than the Yoneda embedding $\mathbf{y} : C \to \mathbf{Set}^{C^{op}}$?

Hence we can work in the category of presheaves $\mathbf{Set}^{C^{op}}$, whose type-theoretic internal language is already well-understood, to study that of C. The universe $\mathcal{U}$ is then given by:

1. an object of $\mathbf{Set}^{C^{op}}$, i.e. a contravariant functorial assignment, to each context Γ, of a set $\mathcal{U}(\Gamma)$ of *types in context* Γ, together with

2. an object $u \in \mathbf{Set}^{C^{op}}/\mathcal{U}$, i.e. a natural transformation $u : \mathcal{U}_\bullet \to \mathcal{U}$, where for each context Γ, $\mathcal{U}_\bullet(\Gamma)$ is the set of terms in context Γ, and $u_\Gamma : \mathcal{U}_\bullet(\Gamma) \to \mathcal{U}(\Gamma)$ assigns each term to its type.

The condition that all types in $\mathcal{U}$ "belong to C" can then be expressed by requiring u to be *representable* in the following sense: for any representable $\gamma \in \mathbf{Set}^{C^{op}}$ with $\alpha : \gamma \to \mathcal{U}$, the pullback

$$\begin{array}{ccc} \gamma.\alpha & \longrightarrow & \mathcal{U}_\bullet \\ \downarrow & \lrcorner & \downarrow u \\ \gamma & \xrightarrow[\alpha]{} & \mathcal{U} \end{array}$$

is representable. In particular, this says that, given a context Γ and a type $A \in \mathcal{U}[\Gamma]$, there is a context Γ, A together with a substitution $\Gamma, A \to A$ that corresponds to the above pullback under the Yoneda embedding. Type-theoretically, this corresponds to the operation of *context extension*.

The question, then, is how to express that C has Σ types, Π types, etc., in terms of the structure of u. Toward answering this question, we may note that u gives rise to an endofunctor (indeed, a *polynomial endofunctor*) $P_u : \mathbf{Set}^{C^{op}} \to \mathbf{Set}^{C^{op}}$, defined by

$$X \mapsto \sum_{A:\mathcal{U}} X^{\mathcal{U}_\bullet[A]}$$

(This notation will be made precise momentarily). As it turns out, much of the type-theoretic structure of u (and by extension C) can be accounted for in terms of this functor. For instance, u is closed under unit and Σ types if and only if $\overline{u}$ carries the structure of a *Cartesian pseudomonad* on $\mathbf{Set}^{C^{op}}$. To see why this is the case, and to expand this account to other type-formers, such as Π types, we will need the theory of polynomial functors on a Locally Cartesian Closed category.

2 Polynomial Arithmetic & Dependent Types

2.1 Polynomial Functors

Recall that a category $\mathcal{C}$ is *Locally Cartesian Closed* (LCC) if $\mathcal{C}$, and all slices of $\mathcal{C}$ are Cartesian closed.

Example: For any category $\mathcal{C}$, $\mathbf{Set}^{\mathcal{C}^{op}}$ is Locally Cartesian Closed.

In particular, if $\mathcal{C}$ is Locally Cartesian Closed, then $\mathcal{C}$ has all pullbacks, and the pullback functors $f^* : \mathcal{C}/\mathrm{A} \to \mathcal{C}/\mathrm{B}$ for any $f : \mathrm{B} \to \mathrm{A}$ have both left and right adjoints Σ_f and Π_f. Hence (up to strictness issues) every LCCC has a form of Martin-Löf Type Theory as its internal type-theoretic language. In what follows, it will therefore be productive to consider polynomial functors and related constructions both as structures *on* an LCCC $\mathcal{C}$, and as structures *in* the internal type-theoretic language of $\mathcal{C}$.

Given any morphism $f : \mathrm{B} \to \mathrm{A} \in \mathcal{C}$, we may define the corresponding *polynomial endofunctor* P_f as the composite

$$\mathcal{C} \simeq \mathcal{C}/1 \xrightarrow{!_{\mathrm{B}}^*} \mathcal{C}/\mathrm{B} \xrightarrow{\Pi_f} \mathcal{C}/\mathrm{A} \xrightarrow{\Sigma_{!_{\mathrm{A}}}} \mathcal{C}/1 \simeq \mathcal{C}$$

Type-theoretically, we think of $f : \mathrm{B} \to \mathrm{A}$ as an A-indexed family $\mathrm{B}[a]$ for all $a : \mathrm{A}$. Then, in the internal language of $\mathcal{C}$, P_f maps a type y to the type

$$\Sigma x : \mathrm{A}.\, y^{\mathrm{B}[x]}$$

In general, polynomial functors P_f can be characterized by the following universal property:

Theorem:

$$\mathrm{H}om_{\mathcal{C}}(\mathrm{C}, \mathrm{P}_f(\mathrm{D})) \cong \sum_{g:\mathrm{C}\to\mathrm{A}} (\mathrm{B}[g] \to \mathrm{D})$$

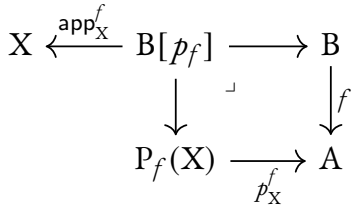
Proof:

$$\begin{aligned} & \mathrm{H}om_{\mathcal{C}}(\mathrm{C}, \mathrm{P}_f(\mathrm{D})) \\ = \;& \mathrm{H}om_{\mathcal{C}}(\mathrm{C}, \Sigma_{!_{\mathrm{A}}} \Pi_f (!_{\mathrm{B}}^*(\mathrm{D}))) \\ \cong \;& \textstyle\sum_{g:\mathrm{C}\to\mathrm{A}} \mathrm{H}om_{\mathcal{C}/\mathrm{A}}(g, \Pi_f(!_{\mathrm{B}}^*(\mathrm{D}))) \\ \cong \;& \textstyle\sum_{g:\mathrm{C}\to\mathrm{A}} \mathrm{H}om_{\mathcal{C}/\mathrm{B}}(f^*(g), !_{\mathrm{B}}^*(\mathrm{D})) \\ \cong \;& \textstyle\sum_{g:\mathrm{C}\to\mathrm{A}} (\mathrm{B}[g] \to \mathrm{D}) \end{aligned}$$

In particular, we obtain natural transformations

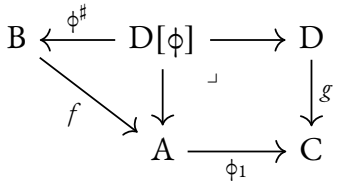
$$p^f : \mathrm{P}_f(-) \Rightarrow \mathrm{A} \quad \text{and} \quad \mathsf{app}^f : \mathrm{B}[p^f_{(-)}] \Rightarrow \mathrm{Id}(-)$$

by plugging in $\mathrm{id}_{\mathrm{P}_f(\mathrm{X})}$ on the right hand side for each $\mathrm{X} \in \mathcal{C}$, yielding following diagram:

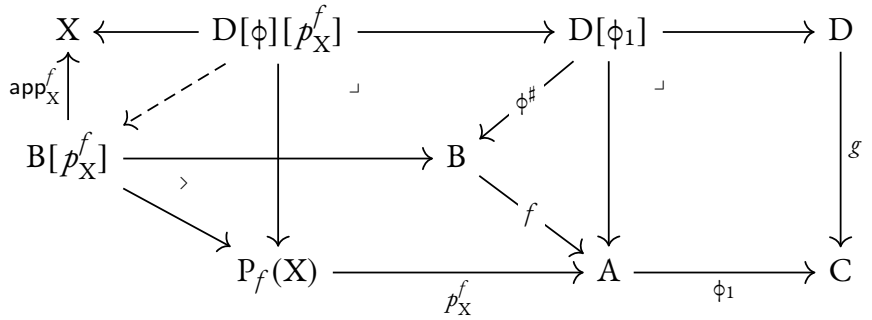


2.1.1 Morphisms of Polynomial Functors

For $f : \mathrm{B} \to \mathrm{A}$ and $g : \mathrm{D} \to \mathrm{C}$, a diagram as below



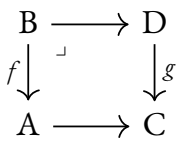
induces a natural transformation $\phi : \mathrm{P}_f \Rightarrow \mathrm{P}_g$ whose components $\phi_{\mathrm{X}} : \mathrm{P}_f(\mathrm{X}) \to \mathrm{P}_g(\mathrm{X})$ are derived by the above equivalence from the following data



One may then wonder: are *all* natural transformations $\mathrm{P}_f \Rightarrow \mathrm{P}_g$ induced in this way? Gambino & Kock (2009) answer this question in the affirmative for *strength-preserving* natural transformations, where polynomial functors are canonically regarded as strong functors in a particular way.

Hence for any locally cartesian closed category $\mathcal{C}$, we have a category $\mathbf{Poly}_{\mathcal{C}}$ whose objects are families $f : \mathrm{B} \to \mathrm{A} \in \mathcal{C}$, regarded as polynomial functors, and whose morphisms are diagrams as above, regarded as (strength-preserving) natural transformations.

Cartesian morphisms of polynomial endofunctors: a morphism of polynomial functors $\phi : \mathrm{P}_f \Rightarrow \mathrm{P}_g$ as above is called *cartesian* if $\phi^\sharp$ is an isomorphism. Equivalently, a Cartesian morphism $\mathrm{P}_f \Rightarrow \mathrm{P}_g$ consists of a pullback square



Hence there is a wide subcategory $\mathbf{Poly}_C^{\mathbf{Cart}} \hookrightarrow \mathbf{Poly}_C$ consisting of polynomial functors and Cartesian morphisms between them.

Type theoretically, thinking of f and g as families of types, the existence of a Cartesian morphism $\phi : \mathrm{P}_f \Rightarrow \mathrm{P}_g$ essentially shows that, for each $a : \mathrm{A}$ there is an element $\phi_1(a) : \mathrm{C}$ such that $\mathrm{B}[a] \simeq \mathrm{D}[\phi_1(a)]$, i.e. every type in the family $\mathrm{B}[a]$ corresponds to a type in the family $\mathrm{D}[c]$. So in particular, a Cartesian morphism in $\mathbf{Set}^{C^{op}}$ from a family $f : \mathrm{B} \to \mathrm{A}$ into a representable family $u : \mathcal{U}_\bullet \to \mathcal{U}$ suffices to show that types in the internal language of C are closed under those types encoded by f. Hence to show that C is closed under Σ types, Π types, etc., we need only find polynomial functors that suitably represent these types, and ask that there be Cartesian morphisms from these polynomials to u.

2.2 Composition of Polynomials & Σ Types

Theorem: Polynomial functors are closed under composition.
Proof: By the universal property of polynomial functors, to show that $\mathrm{P}_f \circ \mathrm{P}_g$ – for $f : \mathrm{B} \to \mathrm{A}$ and $g : \mathrm{D} \to \mathrm{C}$ – is polynomial, it suffices to find $f \triangleleft g : \mathrm{F} \to \mathrm{E}$ such that

$$
\begin{aligned}
& \Sigma_{k:\mathrm{X}\to\mathrm{E}} \left(\mathrm{F}[k] \to \mathrm{Y}\right) \\
\cong\ & \mathrm{H}om(\mathrm{X}, \mathrm{P}_{f\triangleleft g}(\mathrm{Y})) \\
\cong\ & \mathrm{H}om(\mathrm{X}, \mathrm{P}_f(\mathrm{P}_g(\mathrm{Y}))) \\
\cong\ & \Sigma_{k:\mathrm{X}\to\mathrm{A}} \left(\mathrm{B}[k] \to \mathrm{P}_g(\mathrm{Y})\right) \\
\cong\ & \Sigma_{k:\mathrm{X}\to\mathrm{A}} \Sigma_{k':\mathrm{B}\times_{f,k}\mathrm{X}\to\mathrm{C}} \left(\mathrm{D}[k'] \to \mathrm{Y}\right) \\
\cong\ & \Sigma_{k:\mathrm{X}\to\mathrm{A},\ k':\mathrm{B}\times_{f,k}\mathrm{X}\to\mathrm{C}} (\mathrm{D}[k'] \to \mathrm{Y}) \\
\cong\ & \Sigma_{k:\mathrm{X}\to\mathrm{P}_f(\mathrm{C})} \left((\Sigma_{\mathrm{B}[p_{\mathrm{C}}^f]} \mathrm{D}[\mathsf{app}_{\mathrm{C}}^f])[k] \to \mathrm{Y} \right)
\end{aligned}
$$

Hence we can satisfy this by taking $f \triangleleft g$ to be the following

$$
\begin{array}{ccccc}
\mathrm{D} & \longleftarrow & \mathrm{D}[\mathsf{app}_{\mathrm{C}}^f] & & \\
{\scriptstyle g}\downarrow & \lrcorner & \downarrow & & \\
\mathrm{C} & \underset{\mathsf{app}_{\mathrm{C}}^f}{\longleftarrow} & \mathrm{B}[p_{\mathrm{C}}^f] & \longrightarrow & \mathrm{B} \\
 & & \downarrow & \lrcorner & \downarrow{\scriptstyle f} \\
 & & \mathrm{P}_f(\mathrm{C}) & \underset{p_{\mathrm{C}}^f}{\longrightarrow} & \mathrm{A}
\end{array}
$$

It follows that the composition defines a monoidal product on $\mathbf{Poly}_C$, whose unit is the identity functor, represented by the identity morphism $\mathrm{id}_1 : 1 \to 1$ on the terminal object 1.

Type-theoretically, given an A-indexed family of types $\mathrm{B}[a]$ for all $a : \mathrm{A}$ and a C-indexed family of types $\mathrm{D}[c]$ for all $c : \mathrm{C}$, the family representing the composition of B and D, thought

of as polynomial functors, is the $\Sigma x : \mathrm{A}.\mathrm{C}^{\mathrm{B}[x]}$-indexed family that maps a pair $(a, f) : \Sigma x : \mathrm{A}.\mathrm{C}^{\mathrm{B}[x]}$ to the type

$$\Sigma y : \mathrm{B}[a].\mathrm{D}[f(y)]$$

From this follows straightforwardly the universal property of $f \triangleleft g$ as an object of $\mathbf{Poly}_C$:

$$\mathrm{H}om_{\mathbf{Poly}_C}(f \triangleleft g, h) \cong \sum_{\phi_1 : \mathrm{P}_f(\mathrm{C}) \to \mathrm{E}} \mathrm{H}om_{C/\mathrm{P}_f(\mathrm{C})}(\mathrm{F}[\phi_1], \Sigma_{\mathrm{B}[p^f_{\mathrm{C}}]} \mathrm{D}[\mathsf{app}^f_{\mathrm{C}}])$$

and moreover as an object of $\mathbf{Poly}_C^{\mathbf{Cart}}$:

$$\mathrm{H}om_{\mathbf{Poly}_C^{\mathbf{Cart}}}(f \triangleleft g, h) \cong \sum_{\phi_1 : \mathrm{P}_f(\mathrm{C}) \to \mathrm{E}} \left(\mathrm{F}[\phi_1] \cong_{C/\mathrm{P}_f(\mathrm{C})} \Sigma_{\mathrm{B}[p^f_{\mathrm{C}}]} \mathrm{D}[\mathsf{app}^f_{\mathrm{C}}] \right)$$

So in particular, if we take C to be $\mathbf{Set}^{C^{op}}$ and $f = g = h = u$, we have that a Cartesian morphism $\mathrm{P}_{u \triangleleft u} \Rightarrow \mathrm{P}_u$ consists of:

- A natural transformation $\sigma : \left(\sum_{\mathrm{A}:\mathcal{U}} \mathcal{U}^{\mathcal{U}_\bullet[\mathrm{A}]} \right) \to \mathcal{U}$, which assigns to each pair A, B consisting of a type $\mathrm{A} : \mathcal{U}$ and a family of types $\mathrm{B}[a] : \mathcal{U}$ for all $a : \mathcal{U}_\bullet[\mathrm{A}]$, a type $\sigma(\mathrm{A}, \mathrm{B}) : \mathcal{U}$ representing the dependent pair type.
- For each $(\mathrm{A}, \mathrm{B}) : \sum_{\mathrm{A}:\mathcal{U}} \mathcal{U}^{\mathcal{U}_\bullet[\mathrm{A}]}$, a natural transformation $\psi_{\mathrm{A},\mathrm{B}} : \mathcal{U}_\bullet[\sigma(\mathrm{A}, \mathrm{B})] \to \sum_{a:\mathcal{U}_\bullet[\mathrm{A}]} \mathcal{U}_\bullet[\mathrm{B}[a]]$ which maps an element $p : \mathcal{U}_\bullet(\sigma(\mathrm{A}, \mathrm{B}))$ to elements $\pi_1(p) : \mathcal{U}_\bullet[\mathrm{A}]$ and $\pi_2(p) : \mathcal{U}_\bullet[\mathrm{B}[\pi_1(p)]]$, corresponding to the first and second projections out of p, respectively.
- For each A, B as above, a natural transformation $\psi^{-1}_{\mathrm{A},\mathrm{B}} : \left(\sum_{a:\mathcal{U}_\bullet[\mathrm{A}]} \mathcal{U}_\bullet[\mathrm{B}[a]] \right) \to \mathcal{U}_\bullet[\sigma(\mathrm{A}, \mathrm{B})]$, which maps a pair consisting of $a : \mathcal{U}_\bullet[\mathrm{A}]$ and $b : \mathcal{U}_\bullet[\mathrm{B}[a]]$ to an element $\langle a, b \rangle : \mathcal{U}_\bullet[\sigma(\mathrm{A}, \mathrm{B})]$, corresponding to the pairing of a and b.
- Such that $\psi_{\mathrm{A},\mathrm{B}}$ and $\psi^{-1}_{\mathrm{A},\mathrm{B}}$ are mutually inverse.

By inspection, we see that these data correspond precisely to the rules for the Σ type in Martin-Löf type theory. Similarly, u classifies the unit type iff there is a Cartesian morphism $\mathrm{Id}_C \simeq \mathrm{P}_{\mathrm{id}_1} \Rightarrow \mathrm{P}_u$. Hence an object $u \in \mathbf{Poly}$, thought of as a *universe* of types, is closed under unit and Σ types iff there are morphisms

$$\mathrm{id}_1 \to u \qquad u \triangleleft u \to u$$

in $\mathbf{Poly}^{\mathbf{Cart}}$. One might notice, at this point, that these morphisms look suspiciously like the unit and multiplication maps of a monad. Whether or not these operations do in fact define a monad will be our topic for the final section of this post. For now, let us turn our attention to Π types.

2.3 The $\Uparrow$ functor & Π Types

We now wish to define an operation on polynomial functors that type-theoretically corresponds to the formation of Π-types. I.e., in type-theoretic notation, given an A-indexed family $\mathrm{B}[a]$ for all $a : \mathrm{A}$ and a C-indexed family $\mathrm{D}[c]$ for all $c : \mathrm{C}$ – represented by the polynomial functors P_f and P_g for $f : \mathrm{B} \to \mathrm{A}$ and $g : \mathrm{B} \to \mathrm{C}$ – we want to find a polynomial functor $\mathrm{P}_{f \Uparrow g}$ for some $f \Uparrow g : \mathrm{F} \to \mathrm{E}$ that corresponds to the $\Sigma x : \mathrm{A}.\mathrm{C}^{\mathrm{B}[x]}$-indexed family that maps $(a, f) : \Sigma x : \mathrm{A}.\mathrm{C}^{\mathrm{B}[x]}$ to the type

$$\Pi y : \mathrm{B}[a].\mathrm{D}[f(y)]$$

By analogy with the definition of composition in terms of Σ (i.e. $f \triangleleft g = \Sigma_{\mathrm{B}[p_{\mathrm{C}}^f]} \mathrm{D}[\mathsf{app}_{\mathrm{C}}^f] \in C/\mathrm{P}_f(\mathrm{C})$), this should be

$$f \Uparrow g = \Pi_{\mathrm{B}[p_{\mathrm{C}}^f]} \mathrm{D}[\mathsf{app}_{\mathrm{C}}^f] \in C/\mathrm{P}_f(\mathrm{C})$$

Unlike $f \triangleleft g$, however, this definition does not straightforwardly define a functor $\Uparrow : \mathbf{Poly} \times \mathbf{Poly} \to \mathbf{Poly}$. The problem is essentially due to the contravariance of Π in its first argument. If we try to define in type-theoretic notation a functorial action of $\Uparrow$ in its first argument, we face the following problem, given:

- An A-indexed family $\mathrm{B}[a]$ for all $a : \mathrm{A}$
- An A′-indexed family $\mathrm{B}'[a']$ for all $a' : \mathrm{A}'$
- A C-indexed family $\mathrm{D}[c]$ for all $c : \mathrm{C}$
- A function $\phi_1 : \mathrm{A} \to \mathrm{A}'$
- A function $\phi^\sharp : (x : \mathrm{A}) \to \mathrm{B}'[\phi_1(x)] \to \mathrm{B}[x]$

define a function

$$(\phi \Uparrow \mathrm{D})^\sharp : (a : \mathrm{A}, f : \mathrm{C}^{\mathrm{B}[a]}) \to (\Pi b' : \mathrm{B}'[\phi_1(a)].\mathrm{D}[f(\phi^\sharp(a, b'))]) \to \Pi b : \mathrm{B}[a].\mathrm{D}[f(b)].$$

And this is not generally possible, since given $b : \mathrm{B}[a]$ and $g : \Pi b' : \mathrm{B}'[\phi_1(a)].\mathrm{D}[f(\phi^\sharp(a, b'))]$, we have no given way of obtaining a value $b' : \mathrm{B}'[\phi_1(a)]$ from b to supply as input to g.

It *is* possible, however, if we additionally assume that $\phi^\sharp(a, -)$ is invertible for all $a : \mathrm{A}$, i.e. ϕ is Cartesian. Then we can define

$$(\phi \Uparrow \mathrm{D})^\sharp(a, f) = \lambda g.\lambda b.g({\phi^\sharp}^{-1}(a, b))$$

Hence we can view $\Uparrow$ as a functor $\mathbf{Poly}^{\mathbf{Cart}} \times \mathbf{Poly} \to \mathbf{Poly}$.

We can in fact define a more compact representation of $f \Uparrow g$, due to the following theorem:

Theorem: $\Pi_{\mathrm{B}[p^f_{\mathrm{C}}]}\mathrm{D}[\mathsf{app}^f_{\mathrm{C}}] \cong_{C/\mathrm{P}_f(\mathrm{C})} \mathrm{P}_f(g)$

Proof:

$$
\begin{aligned}
& \mathrm{H}om_{C/\mathrm{P}_f(\mathrm{C})}(h, \mathrm{P}_f(g)) \\
\cong & \left\{ k_1 : \mathrm{F} \to \mathrm{A},\ k^\sharp : \mathrm{B}[k_1] \to \mathrm{D} \;\middle|\; \begin{array}{ccccccc} & & & & \mathrm{C} & \xleftarrow{\mathsf{app}^f_{\mathrm{C}}} & \\ & & & \overset{g}{\nearrow} & & & \\ \mathrm{D} & \xleftarrow[k^\sharp]{} & \mathrm{B}[k_1] & \longrightarrow & \mathrm{B}[p^f_{\mathrm{C}}] & \longrightarrow & \mathrm{B} \\ & & \downarrow & & \downarrow & & \downarrow f \\ & & \mathrm{F} & \xrightarrow{h} & \mathrm{P}_f(\mathrm{C}) & \xrightarrow{p^f_{\mathrm{C}}} & \mathrm{A} \end{array} \right\} \\
\cong & \ \mathrm{H}om_{C/\mathrm{C}}(\Sigma_{\mathsf{app}^f_{\mathrm{C}}}(\mathrm{B}[p^f_{\mathrm{C}}][h]), g) \\
\cong & \ \mathrm{H}om_{C/\mathrm{B}[p^f_{\mathrm{C}}]}(\mathrm{B}[p^f_{\mathrm{C}}][h], \mathrm{D}[\mathsf{app}^f_{\mathrm{C}}]) \\
\cong & \ \mathrm{H}om_{C/\mathrm{P}_f(\mathrm{C})}(h, \Pi_{\mathrm{B}[p^f_{\mathrm{C}}]}(\mathrm{D}[\mathsf{app}^f_{\mathrm{C}}]))
\end{aligned}
$$

Using this, we can further characterize $\uparrow\uparrow$ as follows:

Definition: a (left-)*action* of a monoidal category $(\mathcal{M}, \otimes, \mathrm{I})$ on a category C is a functor $\odot : \mathcal{M} \times C \to C$ together with natural isomorphisms:

- $\eta_{\mathrm{A}} : \mathrm{I} \odot \mathrm{A} \cong \mathrm{A}$
- $\mu : (\mathrm{M} \otimes \mathrm{N}) \odot \mathrm{A} \cong \mathrm{M} \odot (\mathrm{N} \odot \mathrm{A})$
- such that the triangles:

$$
\begin{array}{ccccc}
(\mathrm{I} \otimes \mathrm{M}) \odot \mathrm{A} & & & & (\mathrm{M} \otimes \mathrm{I}) \odot \mathrm{A} \\
\downarrow & \searrow & & \swarrow & \downarrow \\
\mathrm{I} \odot (\mathrm{M} \odot \mathrm{A}) & \longrightarrow & \mathrm{M} \odot \mathrm{A} & \longleftarrow & \mathrm{M} \odot (\mathrm{I} \odot \mathrm{A})
\end{array}
$$

and the pentagon

$$
\begin{array}{ccccc}
 & & (\mathrm{M} \otimes (\mathrm{N} \otimes \mathrm{K})) \odot \mathrm{A} & & \\
 & \nearrow & & \searrow & \\
((\mathrm{M} \otimes \mathrm{N}) \otimes \mathrm{K}) \odot \mathrm{A} & & & & \mathrm{M} \odot ((\mathrm{N} \otimes \mathrm{K}) \odot \mathrm{A}) \\
\downarrow & & & & \downarrow \\
(\mathrm{M} \otimes \mathrm{N}) \odot (\mathrm{K} \odot \mathrm{A}) & & \longrightarrow & & \mathrm{M} \odot (\mathrm{N} \odot (\mathrm{K} \odot \mathrm{A}))
\end{array}
$$

formed from these and the associator and unitors of $\mathcal{M}$ all commute.

Theorem: $\Uparrow : \mathbf{Poly}^{\mathbf{Cart}} \times \mathbf{Poly} \to \mathbf{Poly}$ carries the structure of action of the monoidal category $(\mathbf{Poly}^{\mathbf{Cart}}, \triangleleft, \mathrm{id}_1)$ on $\mathbf{Poly}$.

Proof sketch: since

$$f \Uparrow g \cong \mathrm{P}_f(g)$$

we have

- $\mathrm{id}_1 \Uparrow g \cong \mathrm{P}_{\mathrm{id}_1}(g) \cong g$
- $(f \triangleleft g) \Uparrow h \cong \mathrm{P}_{f \triangleleft g}(h) \cong \mathrm{P}_f(\mathrm{P}_g(h)) \cong f \Uparrow (g \Uparrow h)$

and it is straightforward to verify that these satisfy the pentagon and triangle identities.

Finally, to see that $\Uparrow$ has the desired property of classifying Π types on natural models, we first note that the definition of $f \Uparrow g$ as $\Pi_{\mathrm{B}[p^f_{\mathrm{C}}]}\mathrm{D}[\mathsf{app}^f_{\mathrm{C}}]$ straightforwardly yields the following universal property of $f \Uparrow g$ as an object of $\mathbf{Poly}$:

$$\mathrm{H}om_{\mathbf{Poly}_C}(f \Uparrow g, h) \cong \sum_{\phi_1 : \mathrm{P}_f(\mathrm{C}) \to \mathrm{E}} \mathrm{H}om_{C/\mathrm{P}_f(\mathrm{C})}(\mathrm{F}[\phi_1], \Pi_{\mathrm{B}[p^f_{\mathrm{C}}]}\mathrm{D}[\mathsf{app}^f_{\mathrm{C}}])$$

and moreover as an object of $\mathbf{Poly}^{\mathbf{Cart}}_C$:

$$\mathrm{H}om_{\mathbf{Poly}^{\mathbf{Cart}}_C}(f \Uparrow g, h) \cong \sum_{\phi_1 : \mathrm{P}_f(\mathrm{C}) \to \mathrm{E}} \left(\mathrm{F}[\phi_1] \cong_{C/\mathrm{P}_f(\mathrm{C})} \Pi_{\mathrm{B}[p^f_{\mathrm{C}}]}\mathrm{D}[\mathsf{app}^f_{\mathrm{C}}] \right)$$

Then as before, working over $\mathbf{Set}^{C^{op}}$ with $f = g = h = u$, a Cartesian morphism $\mathrm{P}_{u \Uparrow u} \Rightarrow \mathrm{P}_u$ consists of:

- A natural transformation $\pi : \left(\sum_{\mathrm{A}:\mathcal{U}} \mathcal{U}^{\mathcal{U}_\bullet[\mathrm{A}]} \right) \to \mathcal{U}$, which assigns to each pair (A, B) consisting of a type $\mathrm{A} : \mathcal{U}$ and a family of types $\mathrm{B}[a] : \mathcal{U}$ for each $a : \mathcal{U}_\bullet[\mathrm{A}]$, a type $\pi(\mathrm{A}, \mathrm{B}) : \mathcal{U}$, corresponding to the Π type of A and B.
- For each pair A, B as above, a natural transformation $\alpha_{\mathrm{A,B}} : \mathcal{U}_\bullet[\pi(\mathrm{A}, \mathrm{B})] \to \prod_{a:\mathcal{U}_\bullet[\mathrm{A}]} \mathcal{U}_\bullet[\mathrm{B}[a]]$ which, given $f : \mathcal{U}_\bullet[\pi(\mathrm{A}, \mathrm{B})]$ and $a : \mathcal{U}_\bullet[\mathrm{A}]$, produces $\alpha(f)(a) : \mathcal{U}_\bullet[\mathrm{B}[a]]$ corresponding to the result of applying f to a.
- For each pair A, B as above, a natural transformation $\lambda_{\mathrm{A,B}} : \left(\prod_{a:\mathcal{U}_\bullet[\mathrm{A}]} \mathcal{U}_\bullet[\mathrm{B}[a]] \right) \to \mathcal{U}_\bullet[\pi(\mathrm{A}, \mathrm{B})]$ which maps a function $f : \prod_{a:\mathcal{U}_\bullet[\mathrm{A}]} \mathcal{U}_\bullet[\mathrm{B}[a]]$ to a term $\lambda(f) : \mathcal{U}_\bullet[\pi(\mathrm{A}, \mathrm{B})]$ representing the λ-abstraction of f,
- Such that $\alpha_{\mathrm{A,B}}$ and $\lambda_{\mathrm{A,B}}$ are mutually inverse.

Once again, by inspection we see that these data correspond precisely with the rules for Π types in Martin-Löf type theory. Hence we can compactly represent a natural model of MLTT as a representable natural transformation $u : \mathcal{U}_\bullet \to \mathcal{U}$ equipped with morphisms

$$\mathsf{id}_1 \to u \quad u \triangleleft u \to u \quad u \Uparrow u \to u$$

in $\mathbf{Poly}^{\mathbf{Cart}}$. But what sort of structure do these morphisms define on u? Perhaps surprisingly, answering this question satisfactorily requires us to leave the 1-dimensional universe of set theory, and instead reformulate the notion of polynomial universes we have developed in the context of *Homotopy Type Theory* (HoTT).

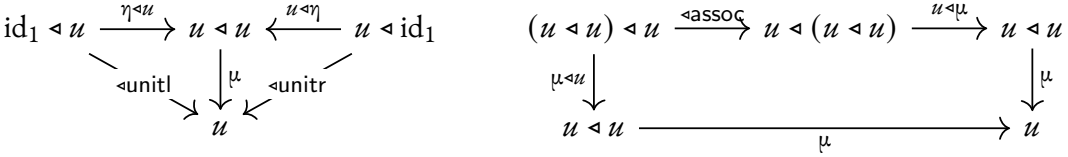
3 Polynomial Universes in HoTT

In order to understand the higher-dimensional identities and coherences of type formers in natural models, we now change our setting from the *extensional* internal language of $\mathbf{Set}^{\mathcal{C}^{op}}$ to the *intensional* internal language of $\infty\mathbf{Grpd}^{\mathcal{C}^{op}}$, which is a form of HoTT. All the constructions on polynomial functors we considered previously carry over to this setting – mutatis mutandis – by replacing the term "category" with "∞-category", etc. Hence we now have ∞-categories **Poly** and $\mathbf{Poly}^{\mathbf{Cart}}$, defined essentially the same as before.

For any polynomial u, we say that u is *univalent* if it is a *subterminal object* in $\mathbf{Poly}^{\mathbf{Cart}}$, i.e. for any other polynomial p, the (homotopy) type of Cartesian morphisms $p \to u$ is a mere proposition in HoTT, i.e. it has at most one inhabitant, up to homotopy.

We call this property of polynomials *univalence* by analogy with the usual univalence axiom. Indeed, the statement that an LCC ∞-Category has enough univalent universes is equivalent to the existence of a *terminal family* in $\mathbf{Poly}^{\mathbf{Cart}}$, i.e. a set of subterminal objects $u_i \in \mathbf{Poly}^{\mathbf{Cart}}$ such that every $p \in \mathbf{Poly}^{\mathbf{Cart}}$ has a Cartesian morphism to *some* u_i.

If u is univalent and has (e.g.) Cartesian morphisms $\eta : \mathsf{id}_1 \to u$ and $\mu : u \triangleleft u \to u$ exhibiting that u is closed under unit and Σ-types, then the following diagrams necessarily commute (up to homotopy)



One may recognize these as the usual diagrams for a monoid in a monoidal category, hence (since $\triangleleft$ corresponds to composition of polynomial endofunctors) for a *monad* as typically defined. Moreover, since the types of Cartesian morphisms into u are contractible, any higher-dimensional diagram one may draw with these also commutes, ensuring that u additionally satisfies all the coherences of an ∞-monad.

For Π-types, the situation is somewhat more complex. What David and I established in our work together is that closure of u under Π-types (in addition to unit and Σ types) corresponds to a special kind of *distributive law* of the above monad over itself.

The notion of a distributive law of one monad over another can be viewed as an answer to the question "when is the composition of two monads also a monad?" Given monads m, n (we may assume, for present purposes, that m, n are polynomial) with unit $\eta_m : \mathsf{id}_1 \to m$ and $\eta_n : \mathsf{id}_1 \to n$ and multiplication $\mu_m : m \triangleleft m \to m$ and $\mu_n : n \triangleleft n \to n$, we can straightforwardly

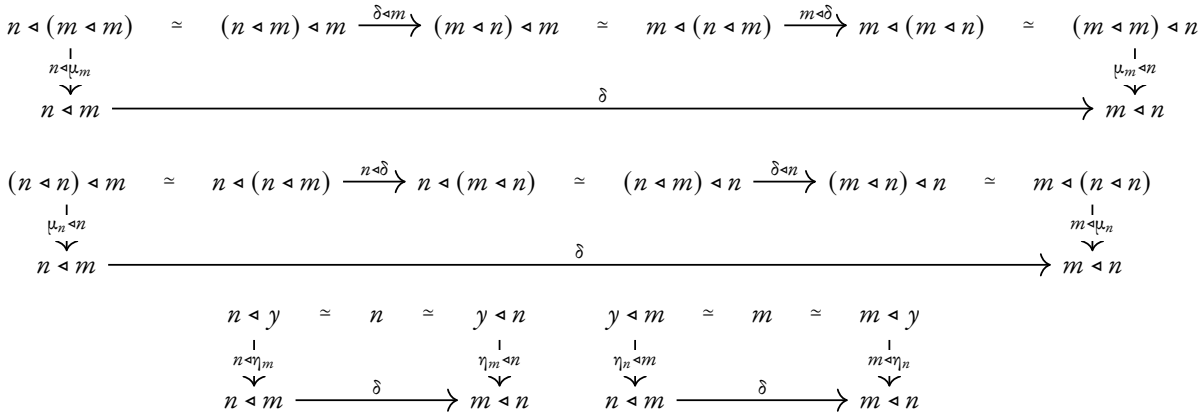
define a putative unit for $m \triangleleft n$ as the composite

$$\mathsf{id}_1 \simeq \mathsf{id}_1 \triangleleft \mathsf{id}_1 \xrightarrow{\eta_m \triangleleft \eta_n} m \triangleleft n$$

However, when we attempt to define a multiplication $(m \triangleleft n) \triangleleft (m \triangleleft n)$, we have no direct way to apply μ_m or μ_n, since neither of the two occurrences of m and n in the domain occur next to each other. This difficulty could be resolved if we had an additional map

$$\delta : n \triangleleft m \to m \triangleleft n$$

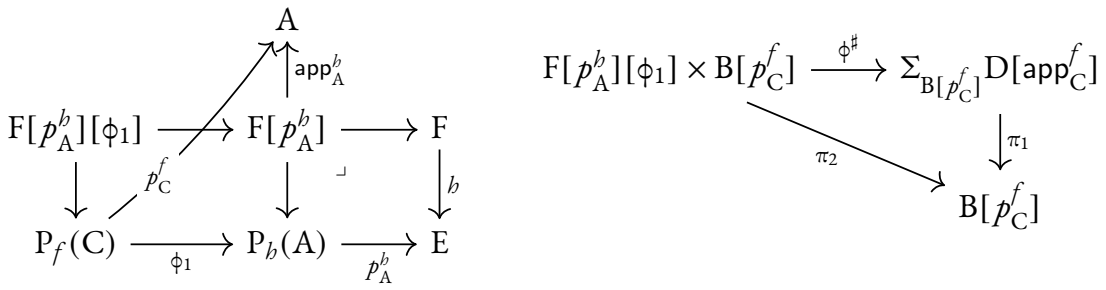
Then indeed, if one reverse-engineers what equations such a map δ must satisfy in order for $m \triangleleft n$ to be a monad with the resulting unit and multiplication, this gives rise to the definition of a *distributive law*. The (1-dimensional) diagrams required to commute in order for δ to be a distributive law are then as follows:



Note that, in the case where $m = n = u$, the morphisms identified by these diagrams are *not* morphisms into u (nor are they generally Cartesian), so we cannot immediately apply univalence to these as we did for the monad laws.

The solution is to define a class of morphisms δ as above that *correspond* to Cartesian morphisms into u.

Definition: a *jump-structure* on a morphism in $\phi : f \triangleleft g \to h \triangleleft f$ in **Poly**, given by $\phi_1 : \mathrm{P}_f(\mathrm{C}) \to \mathrm{P}_h(\mathrm{A})$ and $\phi^\sharp : (\Sigma_{\mathrm{F}[p^h_\mathrm{A}]} \mathrm{B}[\mathsf{app}^h_\mathrm{A}])[\phi_1] \to \Sigma_{\mathrm{B}[p^f_\mathrm{C}]} \mathrm{D}[\mathsf{app}^f_\mathrm{C}]$, consists of paths making the following diagrams commute



essentially, these state that the action of ϕ on its p-components is the identity. This sort of structure was previously considered by David in his blog post on "jump monads," whence the name.

We then leverage the following theorem:

Theorem:

$$\mathrm{H}om_{\mathbf{Poly}}(f \upuparrows g, h) \simeq \mathrm{Jump}(f, g, h)$$

where $\mathrm{Jump}(f, g, h)$ is the type of morphisms $f \triangleleft g \to h \triangleleft g$ equipped with jump structures.

We say that a jump structure is *Cartesian* if its corresponding morphism $f \upuparrows g \to h$ is Cartesian. The proof that a morphism $u \triangleleft u \to u \triangleleft u$ equipped with a Cartesian jump structure automatically satisfies the laws of a (∞-)distributive law then works essentially by showing that Cartesian jump structures are closed under all of the constructions appearing in the diagrams for a distributive law (though note that the definitions/theorems in our paper are a bit more general than what is presented above, in order to accommodate all of the constructions appearing in these diagrams).

Hence, just as a univalent polynomial is closed under Σ and unit types if and only if it possesses the structure of a Cartesian ∞-monad, so too is it closed under Π types if and only if this ∞-monad additionally possesses a distributive law over itself which carries a Cartesian jump structure.

3.1 Examples of polynomial universes

For a univalent universe $\mathcal{U}$, the corresponding polynomial functor looks like

$$\mathrm{X} \mapsto \sum_{\mathrm{A}:\mathcal{U}} \mathrm{X}^{\mathrm{A}}$$

If we specialize this to the case where $\mathcal{U} = \mathbf{Prop}$, the type of propositions, this gives

$$\mathrm{X} \mapsto \sum_{\phi:\mathbf{Prop}} \mathrm{X}^{\phi}$$

This monad is well-known in type theory by another name – the *partiality* monad. Specifically, this is the monad M whose Kleisli morphisms $\mathrm{A} \to \mathrm{M}(\mathrm{B})$ correspond to partial functions $\mathrm{A} \rightharpoonup \mathrm{B}$, i.e. functions that associate to each element $a : \mathrm{A}$ a proposition $\mathsf{def}_f(a)$ stating whether f is defined at input a, such that if $\mathsf{def}_f(a)$ is true, then one can obtain a value $f(a) : \mathrm{B}$.

It follows that one can more generally consider the polynomial monads derived from polynomial universes as *proof-relevant partiality monads*.

3.1.1 Rezk Completion

Additionally, we can show that *any* polynomial functor p can be quotiented to a corresponding univalent polynomial, using a familiar construct from the theory of categories in HoTT – the *Rezk Completion.*

We obtain the Rezk completion of p as the image factorization in $\mathbf{Poly}^{\mathbf{Cart}}$ of the classifying morphism $p \to u_i$ given by the subterminal family mentioned above.

$$p \twoheadrightarrow \mathsf{Rezk}(p) \rightarrowtail u_i$$

and since $\mathsf{Rezk}(p)$ is a subobject of a subterminal object, it follows that it is itself subterminal.

Example: the polynomial functor determined by the function $((m, n) \mapsto n) : \{m < n \in \mathbb{N}\} \to \mathbb{N}$ is

$$\mathrm{X} \mapsto \sum_{n \in \mathbb{N}} \mathrm{X}^{\{m \in \mathbb{N} | m < n\}} \cong \mathsf{List}(\mathrm{X})$$

This polynomial isn't univalent, because $\mathbb{N}$ is a set, whereas the types $\{m \in \mathbb{N} \mid m < n\}$ form a groupoid. However, we can upgrade this to a univalent polynomial using the Rezk completion.

If we write out an explicit description of $\mathsf{Rezk}(\mathsf{List})$, we see that it is the subuniverse of types X that are merely equivalent to some finite type $\{m \in \mathbb{N} \mid m < n\}$. In constructive mathematics, these types (they are necessarily sets) are known as *Bishop finite sets*. Hence the polynomial universe obtained by Rezk completion of the list monad is precisely the subuniverse of types spanned by (Bishop) finite sets.

Question/Food for Thought: Both List and $\mathsf{Rezk}(\mathsf{List})$ are Cartesian monads, hence closed under unit and Σ. However, although List does not have a self-distributive law corresponding to Π types, $\mathsf{Rezk}(\mathsf{List})$ does by the above theorem relating Π types to self-distributive laws of univalent polynomials, since finite sets are closed under finite products. Moreover, although List doesn't *quite* have such a distributive law, it *almost* does, where the pertinent morphism $\mathsf{List} \triangleleft \mathsf{List} \to \mathsf{List} \triangleleft \mathsf{List} \in \mathbf{Poly}$ is the "Cartesian Product" operation on lists that maps a list-of-lists

$$xss = [[x_{11}, \ldots, x_{1i_1}], \ldots, [x_{j1}, \ldots, x_{ji_j}]]$$

to the list of all j-element lists consisting of one element from each list in xss, ordered lexicographically by their occurrence in xss. This operation fails to satisfy some of the equational laws of a distributive law. However, passing to the Rezk completion of List essentially *forces* this operation to satisfy these equations – up to homotopy. It is therefore interesting to consider what structure on a base (non-univalent) polynomial suffices to guarantee that its Rezk completion will be closed under Σ types / Π types.

A potentially related question concerns characterizing the monads of the form $u \triangleleft u$ for u a polynomial universe with Π, Σ, and unit types. Discussions I've had with Steve Awodey, Mathieu Anel & Reid Barton on this point upon my return to CMU this Fall lead me to believe that $\mathsf{Rezk}(\mathsf{List}) \triangleleft \mathsf{Rezk}(\mathsf{List})$ is a kind of higher-dimensional analgoue of the free commutative ring monad – a surprising result, considering that the ordinary, set-based free commutative ring monad is *not* polynomial, whereas $\mathsf{Rezk}(\mathsf{List}) \triangleleft \mathsf{Rezk}(\mathsf{List})$ is. If this result holds, it serves as a further indication that the sheer power of the language of polynomial functors can yet be carried to even greater heights by interpreting this language in the context of HoTT.

4 Conclusion

This post has outlined some of the main results David and I were able to work out during my time at Topos. However, in some sense, this summary barely scratches the surface of how rewarding and intellectually stimulating my time at Topos this Summer truly was. I greatly appreciated the opportunities I had to regularly exchange ideas with David and others at Topos, and to conduct

research on topics of deep interest to me in such an environment. Since my return to CMU, I have continued to build upon the research I conducted at Topos this summer, and I am excited to continue exploring these and related topics further. In particular, I am interested in exploring how the type-theoretic semantics outlined above can be developed and generalized to provide internal languages for various notions of interacting processes and dynamical systems that arise in programming, applied math, etc., and to develop a "logic of systems" around these. As this goal seems closely aligned with the mission and interests of the Topos Institute, I anticipate that pursuing it shall keep me in close contact with Topos well into the future, which, based on my experience this Summer, I very much look forward to!

Lastly, I would like to thank David and all the others I had the pleasure of meeting and interacting with on a daily basis at Topos for helping to make my time there so fulfilling!