THE SYNTHETIC THEORY OF POLYNOMIAL FUNCTORS

DAVID SPIVAK, OWEN LYNCH, REED MULLANIX, SOLOMON BOTHWELL,
AND VERITY SCHEEL

The theory of polynomial functors is a powerful mathematical tool that sees use across a broad range of mathematics, ranging from the very applied (modelling dynamical systems) to the very abstract (constructing models of type theory). However, mechanizing the theory of polynomial functors makes it unwieldy to use; you either forgo all the power of abstraction and work at the level of sets, or are forced to work in a setting entirely without variables, which is mentally taxing, and does not scale well to large expressions. There currently exist some graphical tools such as wiring diagrams that do help the situation, but these only cover a fragment of the theory, and are difficult to integrate into proof assistants.

In light of this, we argue that we ought to be working in the *synthetic* theory of polynomial functors, where polynomials and their morphisms are defined as part of the theory directly. This has several benefits. From a purely ergonomic standpoint, having the theory be aware of the existence of maps of polynomials means that working with them is much more pleasant; gone are the days of massive chains of composites! Furthermore, we are able to exploit the rich structure of polynomials during evaluation.

To back up these claims, we present `PolyTT`, a type-theoretic encoding of the synthetic theory of polynomial functors. It combines Martin Löf Type Theory with a fragment of linear type theory, used for constructing morphisms of polynomials.

1. The Type Theory

We omit the standard rules for Martin Löf Type Theory with Tarski Universes. The first extension we add is a judgement $\Gamma \vdash P\ \mathit{poly}$, which denotes that P is polynomial functor.

$$\textsc{Poly-judgement}\ \frac{\Gamma\ \mathit{ctx}}{\Gamma \vdash P\ \mathit{poly}}$$

There is a single formation rule, which constructs a polynomial functor from a base type $I : \mathsf{U}$ and a family $J : I \to \mathsf{U}$.

$$\textsc{Poly-formation}\ \frac{\Gamma \vdash I\ \mathit{type} \qquad \Gamma, a : A \vdash J\ \mathit{type}}{\Gamma \vdash ((i : I) \times J\ i)\ \mathit{poly}}$$

We also a projection for obtaining the the base type of a polynomial, along with a projection that allows us to get the fibre of the family at a point. These have the expected equations when applied to a concrete polynomial.

$$\textsc{Poly-base}\ \frac{\Gamma \vdash P\ \mathit{poly}}{\Gamma \vdash \mathsf{base}\ P\ \mathit{type}} \qquad \textsc{Poly-fibre}\ \frac{\Gamma \vdash P\ \mathit{poly} \qquad \Gamma \vdash i : \mathsf{base}\ P}{\Gamma \vdash \mathsf{fib}\ P\ i\ \mathit{type}}$$

$$\textsc{Poly-base-decoding}\ \frac{}{\mathsf{base}\ ((a : A) \times B\ a) \equiv A} \qquad \textsc{Poly-fibre-decoding}\ \frac{}{\mathsf{fib}\ ((a : A) \times B\ a)\ i \equiv B\ i}$$

1.1. **Polynomial Maps.** As evidenced above, it is relatively simple to add a type of polynomial functors directly to the theory. In fact, if we have 2 universes $\mathsf{U}_0 : \mathsf{U}_1$ then we can directly define a type of polynomials as $((I : \mathsf{U}_0) \times I \to \mathsf{U}_0) : \mathsf{U}_1$. The complexity of the theory lies in the *maps* of polynomials. In elementary terms, we can define a morphism of polynomials $P \Rightarrow Q$ as a function of type

$$P \Rightarrow Q \coloneqq (i : \mathsf{base}\ P) \to (j : \mathsf{base}\ Q) \times (\mathsf{fib}\ Q\ j \to \mathsf{fib}\ P\ i)$$

Insert a good example of something that's annoying to write.

Provide citation for wiring diagrams.

However, writing morphisms in this form is somewhat tedious and counter-intuitive.

Instead, we opt for a syntax that allows for the construction of the "forward" and "backward" components of a polynomial map simultaneously. To do this, we shall introduce a notion of "obligation" of type A that must be somehow discharged. In a wiring diagram, these obligations correspond to input wires to boxes. This suggests that obligations must be linear; all inputs in a wiring diagram must be provided, and provided *exactly* once. Constructing a map $P \Rightarrow Q$ then consists of constructing a value $q^+ : \mathsf{base}\ Q$ from a value $p^+ : \mathsf{base}\ P$, while also consuming an obligations of type $\mathsf{fib}\ P\ p^+$ and creating an obligation of type $\mathsf{fib}\ Q\ q^+$.

With that high-level picture in our minds, we define a new collection of judgments.

$$\textsc{Hom-ctx-judgment}\ \frac{\Gamma\ \mathit{ctx}}{\Gamma \vdash \Psi\ \mathit{hom\text{-}ctx}}$$

$$\textsc{Resolver-judgment}\ \frac{\Gamma\ \mathit{ctx} \qquad \Gamma \vdash \Psi\ \mathit{hom\text{-}ctx}}{\Gamma \mid \Psi \vdash \pi\ \mathit{resolver}}$$

$$\textsc{Hom-judgment}\ \frac{\Gamma\ \mathit{ctx} \qquad \Gamma \vdash \Psi\ \mathit{hom\text{-}ctx} \qquad \Gamma \vdash Q\ \mathit{poly}}{\Gamma \mid \Psi \vdash \phi\ \mathit{hom}\ \mathrm{Q}}$$

Hom contexts are linear contexts containing obligations that may or may not have been fulfilled.

$$\textsc{Hom-ctx-empty}\ \frac{}{\Gamma \vdash \cdot\ \mathit{hom\text{-}ctx}} \qquad \textsc{Hom-ctx-bind}\ \frac{\Gamma \vdash \Psi\ \mathit{hom\text{-}ctx} \qquad a^- \text{ is a name} \qquad \Gamma \mid \Psi \vdash A\ \mathit{type}}{\Gamma \vdash \Psi, a^- : A\ \mathit{hom\text{-}ctx}}$$

$$\textsc{Hom-ctx-set}\ \frac{\Gamma \vdash \Psi\ \mathit{hom\text{-}ctx} \qquad a^- \text{ is a name} \qquad \Gamma \mid \Psi \vdash A\ \mathit{type} \qquad \Gamma \mid \Psi \vdash v : A}{\Gamma \vdash \Psi, a^- : A \coloneqq v\ \mathit{hom\text{-}ctx}}$$

We omit the judgement signatures and rules for the hom context relative judgements of MLTT; these are identical to the standard ones, with following addition. In order to allow for dependent obligations, we add a notion of "borrowing" from a (potentially unfulfilled) obligation. When the obligation has not yet been fulfilled, we have no equations governing a borrow. If the obligation has been fulfilled, the value of a borrow is equal to the value used to fulfill the obligation.

$$\textsc{Borrow}\ \frac{a^- : A \in \Psi}{\Gamma \mid \Psi \vdash \mathsf{borrow}\ a^- : A} \qquad \textsc{Borrow-value}\ \frac{a^- : A \coloneqq v \in \Psi}{\Gamma \mid \Psi \vdash \mathsf{borrow}\ a^- \equiv v}$$

We proceed by giving rules for resolvers; these consist of a sequence of obligations, and values used to fulfill those obligations. Note that linearity of Ψ is enforced by requiring that we can only fulfill an obligation exactly once.

$$\textsc{Resolver-done}\ \frac{\text{all obligations in } \Psi \text{ are fulfilled}}{\Gamma \mid \Psi \vdash \mathsf{done}\ \mathit{resolver}}$$

$$\textsc{Resolver-fulfill}\ \frac{\Gamma \vdash a^+ : A \qquad \Gamma \mid \Psi_1, a^- : A \coloneqq a^+, \Psi_2 \vdash \pi\ \mathit{resolver}}{\Gamma \mid \Psi_1, a^- : A, \Psi_2 \vdash a^- \coloneqq a^+; \pi\ \mathit{resolver}}$$

We can use this to define hom expressions $\phi\ \mathit{hom}\ \mathrm{Q}$, which denote morphisms *into* some polynomial Q. Much like resolvers, we can also fulfill obligations in a hom expression. The crucial difference is how we terminate a hom expression; we are required to give a $q^+ : \mathsf{base}\ Q$, along with a resolver that is allowed to use reference some $i : \mathsf{fib}\ Q\ q^+$.

$$\textsc{Hom-fulfill}\ \frac{\Gamma \vdash a^+ : A \qquad \Gamma \mid \Psi_1, a^- : A \coloneqq a^+, \Psi_2 \vdash \phi\ \mathit{hom}\ \mathrm{Q}}{\Gamma \mid \Psi_1, a^- : A, \Psi_2 \vdash a^- \coloneqq a^+; \phi\ \mathit{hom}\ \mathrm{Q}}$$

$$\textsc{Hom-done}\ \frac{\Gamma \vdash q^+ : \mathsf{base}\ Q \qquad \Gamma, i : \mathsf{fib}\ Q\ q^+ \mid \Psi \vdash \pi\ \mathit{resolver}}{\Gamma \mid \Psi \vdash (q^+, i.\pi)\ \mathit{hom}\ \mathrm{Q}}$$

With this in place, we can define the type of morphisms, along with the associated introduction and elimination rules.

$$\textsc{Hom-formation}\ \frac{\Gamma \vdash P\ \mathit{poly} \qquad \Gamma \vdash Q\ \mathit{poly}}{\Gamma \vdash P \Rightarrow Q\ \mathit{type}} \qquad \textsc{Hom-intro}\ \frac{\Gamma, p^+ : \mathsf{base}\ P \mid p^- : \mathsf{fib}\ P\ p^+ \vdash \phi\ \mathit{hom}\ \mathrm{Q}}{\Gamma \vdash \lambda\ p^+\ p^-.\ \phi : P \Rightarrow Q}$$

$$\textsc{Hom-elim}\ \frac{\Gamma \vdash f : P \Rightarrow Q \qquad \Gamma \vdash p^+ : \mathsf{base}\ P}{\Gamma \vdash f\ p^+ : \Sigma\ (q^+ : \mathsf{base}\ Q)\ (\mathsf{fib}\ Q\ q^+ \to \mathsf{fib}\ P\ p^+)}$$