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 Lof̈ Type Theory with a fragment of linear type theory, used for constructing morphisms of polynomials.

1. Polynomials

We omit the standard rules for Martin Lof̈ Type Theory with Tarski Universes. For simplicity, we shall assume that the universe $\mathcal{U}$ codes itself. This inconsistency can be resolved by instead using a tower of universes. We start by adding a new judgement $\Gamma \vdash A\ \mathit{Poly}$, which denotes that A is a polynomial functor. It has the following formation rule:

$$\textsc{Judg:Poly-formation}\quad \frac{\Gamma \vdash A\ \mathit{Type} \qquad \Gamma, a : A \vdash B\ \mathit{Type}}{\Gamma \vdash (a : A) \times B\ a\ \mathit{Poly}}$$

We also add the following rules for obtaining the base of a polynomial, along with the fibre of a polynomial at a base.

$$\textsc{Poly-base}\quad \frac{\Gamma \vdash P\ \mathit{Poly}}{\Gamma \vdash \mathit{Base}\ P\ \mathit{Type}} \qquad\qquad \textsc{Poly-fibre}\quad \frac{\Gamma \vdash P\ \mathit{Poly} \qquad \Gamma \vdash i : \mathit{Base}\ P}{\Gamma \vdash \mathit{Fib}\ P\ i\ \mathit{Type}}$$

The base and fibre operations do what one would expect when applied to a concrete polynomial.

$$\textsc{Poly-base-decoding}\quad \frac{}{\mathit{Base}\ (a : A) \times B\ a \equiv A}$$

$$\textsc{Poly-fibre-decoding}\quad \frac{\Gamma \vdash A\ \mathit{Type} \qquad \Gamma, a : A \vdash B\ \mathit{Type} \qquad \Gamma \vdash i : \mathit{Base}\ (a : A) \times B\ a}{\mathit{Fib}\ ((a : A) \times B\ a)\ i \equiv B\ i}$$

2. Morphisms of Polynomials

We proceed by defining a type of morphisms between polynomials.

$$\textsc{Hom-formation}\quad \frac{\Gamma \vdash P\ \mathit{Poly} \qquad \Gamma \vdash Q\ \mathit{Poly}}{\Gamma \vdash P \Rightarrow Q\ \mathit{Type}}$$

2.1. **Hom Environments.**

The introduction and elimination rules for this type are the source of all of the power *and* complexity of the theory, so we will need to build out some machinery before we present them. To start, we define hom environments.

$$\textsc{Hom-env-empty}\quad \frac{}{\Gamma \vdash \cdot\ \mathit{HomEnv}} \qquad \textsc{Hom-env-bind}\quad \frac{\Gamma \vdash \rho\ \mathit{HomEnv} \qquad a^{-} \text{ is a name}}{\Gamma \vdash \rho, a^{-}\ \mathit{HomEnv}}$$

$$\textsc{Hom-env-set}\quad \frac{\Gamma \vdash \rho\ \mathit{HomEnv} \qquad \Gamma \mid \rho \vdash v : A \qquad a^{-} \text{ is a name}}{\Gamma \vdash \rho, a^{-} := v\ \mathit{HomEnv}}$$

We shall gloss over any difficulties with name freshness, and gesture vaguely in the direction of nameless representations if pressed. We also define a meta-operation $\rho[a^{-} := v]$, which replaces an a^{-} in an environment ρ with $a^{-} := v$. This is defined by induction over the environment. Furthermore, note that the rule for setting a binding in an environment ρ does *not* allow for the the value v to depend on earlier entries in ρ.

We now define all of the normal judgments relative to some environment ρ.

$$\text{Judg:}\textsc{Type-in-env}\quad \frac{\Gamma\ \mathit{Ctx} \qquad \Gamma \vdash \rho\ \mathit{HomEnv}}{\Gamma \mid \rho \vdash A^{+}\ \mathit{Type}} \qquad \text{Judg:}\textsc{Term-in-env}\quad \frac{\Gamma\ \mathit{Ctx} \qquad \Gamma \vdash \rho\ \mathit{HomEnv} \qquad \Gamma \mid \rho \vdash A^{+}\ \mathit{Type}}{\Gamma \mid \rho \vdash a : A^{+}}$$

$$\text{Judg:}\textsc{Type-eq-in-env}\quad \frac{\Gamma\ \mathit{Ctx} \qquad \Gamma \vdash \rho\ \mathit{HomEnv} \qquad \Gamma \mid \rho \vdash A^{+}\ \mathit{Type} \qquad \Gamma \mid \rho \vdash B^{+}\ \mathit{Type}}{\Gamma \mid \rho \vdash A^{+} \equiv B^{+}\ \mathit{Type}}$$

$$\text{Judg:}\textsc{Term-eq-in-env}\quad \frac{\begin{array}{c}\Gamma\ \mathit{Ctx} \\ \Gamma \vdash \rho\ \mathit{HomEnv} \qquad \Gamma \mid \rho \vdash A\ \mathit{Type} \qquad \Gamma \mid \rho \vdash x\ \mathit{Type} : A^{+} \qquad \Gamma \mid \rho \vdash y\ \mathit{Type} : A^{+}\end{array}}{\Gamma \mid \rho \vdash x \equiv y : A^{+}}$$

We can read variables from an environment.

$$\textsc{Hom-env-read}\quad \frac{a^- \in \rho}{\Gamma \mid \rho \vdash \mathit{read}\ a^- : A^+}$$

All of the rules for types in the base theory are copied over to the environment relative ones. We shall omit these for brevity, though we shall highlight the fact that we have an environment relative conversion rule.

$$\textsc{Conversion-in-env}\quad \frac{\begin{array}{c}\Gamma\ \mathit{Ctx} \qquad \Gamma \vdash \rho\ \mathit{HomEnv}\\ \Gamma \mid \rho \vdash A\ \mathit{Type} \qquad \Gamma \mid \rho \vdash B\ \mathit{Type} \qquad \Gamma \mid \rho \vdash A \equiv B\ \mathit{Type} \qquad \Gamma \mid \rho \vdash x : B\end{array}}{\Gamma \mid \rho \vdash x : A}$$

2.2. **Hom Contexts and Sinks.** We now give rules for linear Hom contexts, as well as a judgement for deriving "sink" terms from linear contexts.

Reed: I *think* it's ok to depend on reads from earlier in ρ?

$$\textsc{Hom-context-empty}\quad \frac{}{\Gamma \mid \rho \vdash \cdot\ \mathit{HomCtx}} \qquad \textsc{Hom-context-bind}\quad \frac{\Gamma \mid \rho \vdash \Psi\ \mathit{HomCtx} \qquad \Gamma \mid \rho \vdash A\ \mathit{Type}}{\Gamma \mid \rho \vdash \Psi, a^- : A\ \mathit{HomCtx}}$$

$$\textsc{Judg:Sink-term}\quad \frac{\begin{array}{c}\Gamma\ \mathit{Ctx}\\ \Gamma \vdash \rho\ \mathit{HomEnv} \qquad \Gamma \mid \rho \vdash \Psi\ \mathit{HomCtx} \qquad \Gamma \mid \rho \vdash A\ \mathit{Type} \qquad \Gamma \vdash \theta\ \mathit{HomEnv}\end{array}}{\Gamma \mid \rho \mid \Psi \vdash a^- : A^- \rightsquigarrow \theta}$$

$$\textsc{Sink-conversion}\quad \frac{\Gamma \mid \rho \mid \Psi \vdash a^- : A^- \qquad \Gamma \mid \rho \vdash A \equiv B\ \mathit{Type}}{\Gamma \mid \rho \mid \Psi \vdash a^- : B^-}$$

Linearity of the Hom contexts is ensured by the following rule. The intuition here is that in order to discharge an obligation a^-, we are required to provide it with a value of the appropriate type.

$$\textsc{Sink-linear-var}\quad \frac{\Gamma \vdash a^+ : A}{\Gamma \mid \rho \mid a^- : A^- \vdash a^-[a^+] : A^- \rightsquigarrow a^- := a^+}$$

2.3. **Hom Expressions.** We proceed by defining a new judgement describing hom expressions.

JUDG:HOM-TERM

$$\frac{\Gamma\ \mathit{Ctx} \qquad \Gamma \vdash \rho\ \mathit{HomEnv} \qquad \Gamma \mid \rho \vdash \Psi\ \mathit{HomCtx} \qquad \Gamma \vdash Q\ \mathit{Poly}}{\Gamma \mid \rho \mid \Psi \vdash \phi\ \mathit{Hom}(Q)}$$

HOM-SET

$$\frac{\Gamma \vdash a^+ : A \qquad \Gamma, x : A \mid \rho \mid \Psi_1 \vdash b^- : B^- \rightsquigarrow \theta \qquad \Gamma, \rho, \theta[a^+/x], \Psi_2 \vdash \phi\ \mathit{Hom}(Q)}{\Gamma \mid \rho \mid \Psi_1, \Psi_2 \vdash a^+ \to (\lambda x.b^-); \phi\ \mathit{Hom}(Q)}$$

HOM-DONE

$$\frac{\Gamma \vdash q^+ : \mathit{Base}\ Q \qquad \Gamma, x : \mathit{Fib}\ Q\ q^+ \mid \rho \mid \Psi \vdash q^- : A^-}{\Gamma \mid \rho \mid \Psi \vdash (q^+, \lambda x.q^-)\ \mathit{Hom}(Q)}$$

Note that we lambda abstract a variable x when defining the sink expression. This is not essential, but allows us to avoid writing a point-free mess. This also removes the need for a separate sink application rule, as it can be handled by function application.

2.4. Hom Introduction and Elimination.

HOM-INTRO

$$\frac{\Gamma, p^+ : \mathit{Base}\ p \mid \cdot \mid p^- : \mathit{Fib}\ P\ p^+ \vdash \phi\ \mathit{Hom}(Q)}{\Gamma \vdash \lambda p^+ p^- \rightsquigarrow \phi : P \Rightarrow Q}$$

HOM-ELIM

$$\frac{\Gamma \vdash f : P \Rightarrow Q \qquad \Gamma \vdash p^+ : \mathit{Base}\ P}{\Gamma \vdash f\ p^+ : (q^+ : \mathit{Base}\ Q) \times (\mathit{Fib}\ Q\ q^+ \to \mathit{Fib}\ P\ p^+)}$$

3. Sink Introduction and Elimination

We need to also have rules for eliminating sink values in hom expressions.

SIGMA-SINK-ELIM

$$\frac{\begin{array}{c}\Gamma, x : (a : A) \times B\ a \mid \rho \mid \Psi_1 \vdash c^- : C^- \rightsquigarrow \theta \\ \Gamma \mid \rho, a^-, b^-, \theta[(\mathit{read}\ a^-, \mathit{read}\ b^-)/x] \mid \Psi_2, a^- : A, b^- : B(\mathit{read}\ a^-) \vdash \phi\ \mathit{Hom}(Q)\end{array}}{\Gamma \mid \rho \mid \Psi_1 \vdash \mathsf{unpack}\ ab^- \to (a^-, b^-); \phi\ \mathit{Hom}(Q)}$$

We also need rules for introducing sink values in sink expressions.

SIGMA-SINK-INTRO

$$\frac{\Gamma \mid \rho \mid \Psi_1 \vdash a^- : A^- \rightsquigarrow \theta_1 \qquad \Gamma \mid \rho \mid \Psi_1 \vdash b^- : B^-(\mathit{read}\ \theta_1) \rightsquigarrow \theta_2}{\Gamma \mid \rho \mid \Psi_1, \Psi_2 \vdash (a^-, b^-) : ((x : A) \times B\ x)^- \rightsquigarrow \theta_1, \theta_2}$$

This is pretty janky; do we have to care about associativity of things?

Here, *read* (θ_1) denotes reading all bindings from θ_1, and tupling them up in the form required by B^-.

We also have rules for introducing values of the unit type in sink expressions. This is mostly relevant when providing the final sink value to a hom expression, as it lets us discard the input entirely.

UNIT-SINK-INTRO

$$\frac{}{\Gamma \mid \rho \mid \cdot \vdash \mathsf{drop} : \top^- \rightsquigarrow \cdot}$$

An elimination rule is not required, as we have η for the unit type, and conversion for sink values allows for the η to apply.