Polynomial Universes and Dependent Types

C.B. Aberlé David I. Spivak

Abstract

Awodey, later with Newstead, showed how polynomial pseudomonads $(u, 1, \Sigma)$ with extra structure (termed "natural models" by Awodey) hold within them the syntax and rules for dependent type theory. Their work presented these ideas clearly but ultimately led them outside of the category of polynomial functors in order to explicate all of the structure possessed by such models of type theory.

This paper builds off that work—explicating the syntax and rules for dependent type theory by axiomatizing them *entirely* in the language of polynomial functors. In order to handle the higher-categorical coherences required by such an explication, we work with polynomial functors internally in the language of Homotopy Type Theory, which allows for higher-dimensional structures such as pseudomonads, etc. to be expressed purely in terms of the structure of a suitably-chosen $(2, 1)$-category of polynomial functors. The move from set theory to Homotopy Type Theory thus has a twofold effect of enabling a simpler exposition of natural models, which is at the same time amenable to formalization in a proof assistant, such as Agda.

Moreover, the choice to remain firmly within the setting of polynomial functors reveals many additional structures of natural models that were otherwise left implicit or not considered by Awodey & Newstead. Chief among these, we highlight the fact that every polynomial pseudomonad $(u, 1, \Sigma)$ as above that is also equipped with structure to interpret dependent product types gives rise to a self-distributive law $u \triangleleft u \to u \triangleleft u$, which witnesses the usual distributive law of dependent products over dependent sums.

1 Introduction

Dependent type theory [Mar75] was founded by Per Martin-Löf in 1975 to formalize constructive mathematics. The basic idea is that the *order of events* is fundamental to the mathematical story arc: when playing out any specific example story in that arc, the beginning of the story affects not only the later events, but even the very terms with which the later events will be described. For example, in the story arc of conditional probability, one may say "now if the set P that we are asked to condition on happens to have measure zero, we must stop; but assuming that's not the case then the result will be a new probability measure.'' Here the story teller is saying that no terms will describe what happens if P has measure zero, whereas otherwise the terms of standard probability will apply.

Dependent types form a logical system with syntax, rules of computation, and methods of deduction. In [Awo14; AN18], Awodey and later Newstead show that there is a strong connection between dependent type theory and polynomial functors, via their concept of *natural models*, which cleanly solve the problem of *strictififying* certain identities that typically hold only up to isomorphism in arbitrary categories, but must hold *strictly* in order for these to soundly model dependent type theory. The solution to this problem offered by Awodey and Newstead makes use of the type-theoretic concept of a *universe*. Such universes then turn out to naturally be regarded as polynomial functors on a suitably-chosen category of presheaves, satisfying a certain *representability* condition.

Although the elementary structure of natural models is thus straightforwardly described by considering them as objects in the category of polynomial functors, Awodey and Newstead were ultimately led outside of this category in order to fully explicate those parts of natural models that require identities to hold only *up to isomorphism*, rather than strictly. There is thus an evident tension between *strict* and *weak* identities that has not yet been fully resolved in the story of natural models. In the present work, we build on Awodey and Newstead's work to fully resolve this impasse by showing how type universes can be fully axiomatized in terms of polynomial functors, by working with polynomial functors internally in the language of *Homotopy Type Theory* (HoTT). We thus come full circle from Awodey's original motivation to develop natural models *of* Homotopy Type Theory, to describing natural models *in* Homotopy Type Theory.

The ability for us to tell the story of natural models as set entirely in the category of polynomial functors has a great simplifying effect upon the resultant theory, and reveals many additional structures, both of polynomial universes, and of the category of polynomial functors as a whole. As an illustration of this, we show how every polynomial universe, regarded as a polynomial pseudomonad with additional structure, gives rise to self-distributive law $u \triangleleft u \longrightarrow u \triangleleft u$, which witnesses the usual distributive law of dependent products over dependent sums.

Moreover, the move from set theory to HoTT as a setting in which to tell this story enables new tools to be applied for its telling. In particular, the account of polynomial universes we develop is well-suited to formalization in a proof assistant, and we present such a formalization in Agda. This paper is thus itself a literate Agda document in which all results have been fully formalized and checked for validity.

```
module poly-universes where

open import Agda.Primitive
open import Agda.Builtin.Equality
open import Agda.Builtin.Sigma
```

The structure of this paper is as follows:

2 Background on HoTT and Polynomial Functors

We begin with a quick recap of the basics of HoTT and polynomial functors.

2.1 Homotopy Type Theory

```
Type : (ℓ : Level) → Set (lsuc ℓ)
Type ℓ = Set ℓ

isContr : ∀ {ℓ} → Type ℓ → Type ℓ
isContr A = Σ A (λ a → (b : A) → a ≡ b)

isProp : ∀ {ℓ} → Type ℓ → Type ℓ
isProp A = (a b : A) → a ≡ b

isSet : ∀ {ℓ} → Type ℓ → Type ℓ
isSet A = (a b : A) → isProp (a ≡ b)

isGrpd : ∀ {ℓ} → Type ℓ → Type ℓ
isGrpd A = (a b : A) → isSet (a ≡ b)

isEquiv : ∀ {ℓ κ} {A : Type ℓ} {B : Type κ}
          → (A → B) → Set (ℓ ⊔ κ)
isEquiv {A = A} {B = B} f = (b : B) → isContr (Σ A (λ a → f a ≡ b))
```

2.2 Polynomials in HoTT

```
PrePoly : (ℓ κ : Level) → Type ((lsuc ℓ) ⊔ (lsuc κ))
PrePoly ℓ κ = Σ (Set ℓ) (λ A → A → Set κ)

Lens : ∀ {ℓ ℓ' κ κ'} → PrePoly ℓ κ → PrePoly ℓ' κ' → Type (ℓ ⊔ ℓ' ⊔ κ ⊔ κ')
Lens (A , B) (C , D) = Σ (A → C) (λ f → (a : A) → D (f a) → B a)

record Poly {ℓ κ} : Type ((lsuc ℓ) ⊔ (lsuc κ)) where
    constructor poly
    field
        Pos : Type ℓ
        Dir : Pos → Type κ
        posIsGrpd : isGrpd Pos
        dirIsSet : (p : Pos) → isSet (Dir p)
```

2.2.1 Basics

2.2.2 Univalence

```
idToEquiv : ∀ {ℓ κ} (p : PrePoly ℓ κ) (a b : fst p)
            → a ≡ b → Σ (snd p a → snd p b) isEquiv
idToEquiv p a b refl = (λ x → x) , (λ y → (y , refl) , (λ { (y' , refl) → refl }))

isUnivalent : ∀ {ℓ κ} → PrePoly ℓ κ → Type (ℓ ⊔ κ)
```

```
isUnivalent (A , B) =
    (a b : A) → isEquiv (idToEquiv (A , B) a b)
```

2.2.3 The Vertical-Cartesian Factorization System

```
isVertical : ∀ {ℓ ℓ' κ κ'} {p : PrePoly ℓ κ} {q : PrePoly ℓ' κ'}
             → Lens p q → Set (ℓ ⊔ ℓ')
isVertical (f , g) = isEquiv f

isCartesian : ∀ {ℓ ℓ' κ κ'} {p : PrePoly ℓ κ} {q : PrePoly ℓ' κ'}
              → Lens p q → Set (ℓ ⊔ κ ⊔ κ')
isCartesian {p = (A , B)} (f , g) = (a : A) → isEquiv (g a)
```

2.2.4 The Composition and Dirichlet Monoidal Products

```
_◃_ : ∀ {ℓ ℓ' κ κ'} → PrePoly ℓ κ → PrePoly ℓ' κ' → PrePoly (ℓ ⊔ κ ⊔ ℓ') (κ ⊔ κ')
(A , B) ◃ (C , D) = (Σ A (λ a → B a → C) , λ (a , f) → Σ (B a) (λ b → D (f b)))

_⊗_ : ∀ {ℓ ℓ' κ κ'} → PrePoly ℓ κ → PrePoly ℓ' κ' → PrePoly (ℓ ⊔ ℓ') (κ ⊔ κ')
(A , B) ⊗ (C , D) = (Σ A (λ _ → C) , λ (a , c) → Σ (B a) (λ _ → D c))
```

2.2.5 The ↑↑ and $(-)^{=}$ Functors

3 Polynomial Universes & Jump Monads

3.1 Polynomial Universes

3.2 Jump Monads & Distributive Laws