Polynomial universes and dependent types

David I. Spivak

Abstract

Awodey, later with Newstead, showed how polynomial monads $(u, 1, \Sigma)$ with extra structure hold within them the syntax and rules for dependent type theory. Their work presented these ideas cleanly but within a complex setting: using pseudomonads and pseudo-algebras in a tri-category.

This paper builds off that work—explicating the syntax and rules for dependent type theory by axiomatizing them in the language of polynomial functors—but from a different starting point. First we work in a more basic setting, that of a 1-category with two monoidal products. Second, rather than considering pseudo-algebras of any sort, we define a seemingly new categorical structure that houses the axioms for dependent type theory in a polynomial setting. This structure picks up something that Awodey-Newstead seem to have missed: the distributive law for products over sums. Indeed, one result of our axiomatization is that any universe monad will always carry a self-distributive law $u \triangleleft u \to u \triangleleft u$.

Mistake: the distributive law

$$\Pi(A, \Sigma(B, C)) \cong \Sigma(\Pi(A, B), \Pi(A, C))$$

does not work. You can't seem to define $u \triangleleft u \to u \otimes u$ in such a way that (13) holds. Here's a counterexample:

$$\begin{array}{ccccc} [[[1,2],[3]],[[4],[5]]] & \longmapsto & [[[1,2],[4]],[[1,2],[5]],[[3],[4]],[[3],[5]]] & \longmapsto & [[[1,4],[2,4]],[[1,5],[2,5]],[[3,4]],[[3,5]]] \\ \downarrow & & & & \downarrow \\ [[1,2,3],[4,5]] & \longmapsto & [[1,4],[2,4],[3,4],[1,5],[2,5],[3,5]] & \neq & [[1,4],[2,4],[1,5],[2,5],[3,4],[3,5]] \end{array}$$

You could try using "the other order", but that doesn't work either.

1 Introduction

Dependent type theory [Mar75] was founded by Per Martin-Löf in 1975 to formalize constructive mathematics. The basic idea is that *order of events* is fundamental to a 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, conversion rules, 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. The present work follows from this remarkable discovery, but diverges in the formalism itself. Whereas they discuss pseudomonads and pseudo-algebras in a tri-category arising from a locally cartesian closed category, we keep to monoidal 1-categories.

A polynomial functor $p\colon \mathbf{Set} \to \mathbf{Set}$ is a coproduct of representable functors

$$\sum_{I \in p(1)} y^{p[I]},$$

and one can think of p as a collection of types: A position $I \in p(1)$ indexing the coproduct corresponds to a *type* in the collection, and an element $i \in p[I]$ in the representing set $p[I]$ corresponds to a *term* of type I.[1]

Under this system, composition $p \triangleleft q$ of polynomials corresponds to dependent types: in order to choose a type $x \in (p \triangleleft q)(1)$ in the composite, you first choose a type $I \in p(1)$ and then, for any term $i \in p[I]$ of it you choose a type $J : q(1)$. In this paper (not in the literature) we denote this situation–where J depends on a choice of term in I—explicitly by the symbol $I \prec J$.

For example, alluding to the probability example in the first paragraph, one may briefly consider the following complicated-looking polynomials:

$$p = \sum_{R=0} y^0 + \sum_{R \in \mathbb{R}_{>0}} y^1 \qquad \text{and} \qquad q = \sum_{N \in \mathbb{N}} \sum_{\{D\colon N \to \mathbb{R}_{\geq 0} \mid D_1 + \cdots + D_N = 1\}} y^N.$$

To choose an element in $(p \triangleleft q)(1)$, you *first* choose a nonnegative real $R \in \mathbb{R}_{\geq 0}$; that's a type in the p-family. Then the type from q depends on your choice. Indeed, in the $(R = 0)$-case you have already hit the end of the story—you stop—and in the $(R > 0)$-case you *further* make choice of $N \in \mathbb{N}$ and a probability distribution D on a set with N elements; that's a type in the q-family. All together, whichever branch you took, the result is a type $R \prec (N, D)$ in the collection $p \triangleleft q$. This type has *no terms* in the $(R = 0)$-case, and it has N-many terms in the $(R > 0)$-case.

A special case of dependent types is *independent* types, where the order of some part of the story *doesn't* matter. We represent these using a symmetric monoidal product $\otimes$ on polynomial functors: for any p, q one can form $p \otimes q$ and there is an isomorphism $p \otimes q \cong q \otimes p$. The types in the family $p \otimes q$ are pairs of types $(I, J) \in p(1) \times q(1)$, and the terms of type (I, J) are pairs of terms $(i, j) \in p[I] \times q[J]$. There is a natural map $\mathit{Indep}\colon (p \otimes q) \to (p \triangleleft q)$ allowing us to regard a pair (I, J) of independent types as a dependent type $(I \prec J) \coloneqq \mathit{Indep}(I, J)$.

[1] This paper is fairly self-contained. We will review polynomial functors in Section 2.

In this paper we explain the notion of a *polynomial universe* u,[2] with which one can reduce all the dependencies by collapsing an arbitrarily deep dependent type into a single layer. In particular we will be interested in a particular map

$$\pi_{p,y} \colon p \triangleleft u \longrightarrow u \otimes p \tag{1}$$

that converts dependent types into independent types by sending $I \prec A$ to the pair $(I \longrightarrow A, I)$, where $I \longrightarrow A$ might be called the *dependent product type* or *dependent function type*.[3] It is as though p jumps over u—skips ahead in line—and in the process causes u to absorb the effect of the old p-dependencies into itself. This is a move which in some sense makes computations into "first-class objects". When $p = u$ it says we can convert a dependent type $(A \prec B) \in u \triangleleft u(1)$ to a base type $(A \longrightarrow B, A) \in u(1)$ by first absorbing the dependency and then composing with a pairing operation $u \triangleleft u \xrightarrow{\pi_{u,y}} u \otimes u \xrightarrow{(-,-)} u$.

In the text below we will give examples of polynomial universes; probably the best known is the $\mathbb{N}$-list monad

$$u_{\mathbb{N}} \coloneqq \sum_{N \in \mathbb{N}} y^N$$

which has a notion of singleton list and a way to concatenate lists of lists into a list

$$[[1,2,3],[4,5]] \quad \mapsto \quad [1,2,3,4,5].$$

But one can also perform the following operation: given a list of lists, one can *list out* all the ways to choose one element from each, e.g.

$$[[1,2,3],[4,5]] \quad \mapsto \quad [[1,4],[1,5],[2,4],[2,5],[3,4],[3,5]].$$

Let's call this operation *cross-sectioning*. Anyway, the ways these two operations—concatenating and cross-sectioning—work together coherently with various associative, distributive, unital, and other properties. The coherence between them is at the heart of dependent type theory and emerges directly from Definition 3.1. It holds for other sets κ in place of $\mathbb{N}$, whenever they have certain properties; we refer to the resulting polynomials u_κ as *κ-list polynomials* and show that each is a polynomial universe.

In this paper we will see that syntax and conversion rules of dependent types emerge from the very special properties enjoyed by polynomial universes u. Not only should a polynomial universe carry the structure of a cartesian monad, which gives it a unit type 1 and Σ types, but these must distribute properly over function types.

The axioms we present for universes are category-theoretically reasonable, and yet at this time we do not have a neat packaging for them. We give a list of six commutative diagrams in Definition 3.1. Together these axioms imply the dependent type syntax and rules we want. They can be packaged up in various ways, though

[2] Again, this notion is due to [Awo14, Theorem 16], where he might call them "natural model of extensional Martin-Löf type theory with product and sum types". We focus on what that paper would call the $\mathbb{C} = 1$ case, meaning that everything in this paper takes place in **Set** rather than an arbitrary presheaf category.

[3] This is just a terminological preference: a product $\prod_{a \in A} B$ of A-many copies of B is also a function $A \longrightarrow B$; in both cases the number of B's could depend on the choice of $a \in A$. Allowing ones terminology to depend on ones preferences is a meta-version of this whole story.

none seems to capture the full strength of the axioms. One packaging is as a pair of lax monoidal functors $\mathbf{Poly}^{\mathbf{Cart}} \to \mathbf{End}(\mathbf{Poly})$ and a lax transformation between them; see Definition 3.2. Another is as a self-distributivity law $u \triangleleft u \to u \triangleleft u$, as discussed in a July 2021 Topos Institute blog post; see Definition 3.3.

As mentioned there, the germination of this idea came from Steve Awodey's talk at the 2021 Workshop on Polynomial Functors on his theory of natural models and their connection to polynomial functors. Aside from a very different presentation (e.g. 1-categories here vs. tri-categories there), the present work adds two things:

1. explicit syntax and rules for independent pair types,[4] and
2. the distributive law of Π over Σ, which in the language of [AN18] would say[5]

$$\prod_{x:A} \sum_{y:B(x)} C(x,y) \cong \sum_{y:\prod_{x:A} B(x)} \prod_{x:A} C(x, y(x)). \tag{2}$$

Awodey and Newstead derived the other law connecting Σ and Π in their paper (see [AN18, Remark 4.2]), but under our reading of it, they seem to be missing the one in (2).[6]

1.1 Acknowledgments

I greatly appreciate the clarity of delivery and the insight behind Steve Awodey's talk at the 2021 Workshop on Polynomial Functors. It has been a major source of inspiration and fun during the intervening year. Thanks to David Jaz Myers for helpful conversations.

This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-20-1-0348.

2 Background and key examples

The main purpose of this section is to fix notation and provide a brief overview of polynomial functors in one variable. More extensive background material can be found in [**spivak2022poly**] and [GK12]. We will also use the present section to introduce our main polynomials of interest, which we call κ-list polynomials.

[4] Explicit syntax for independent pair types can be useful, since the symmetry gives extra flexibility.

[5] In our language it would say

$$A \to \Sigma(B, C) \cong \Sigma(A \to B, A \to C).$$

[6] André Joyal's question following Steve Awodey's talk was "what happened to the distributivity law that we love between sum and product?" Steve said that he believed it was of the same form as the other laws they had derived, and that they get it for free. I have not seen this derivation, at least not in a way I could understand, but this may very well be merely a result of my own inadequacy or lack of appropriate effort to understand what is there.

2.1 Basics

Definition 2.1 (Polynomial functor)**.** Given a set S, we denote the corresponding representable functor by

$$y^S \coloneqq \mathbf{Set}(S, -)\colon \mathbf{Set} \to \mathbf{Set},$$

e.g. $y^S(X) \coloneqq X^S$. In particular $y = y^1$ is the identity and $y^0 = 1$ is constant singleton.

A *polynomial functor* is a functor $p\colon \mathbf{Set} \to \mathbf{Set}$ that is isomorphic to a sum of representables, i.e. for which there exists a set T, a set $p[t] \in \mathbf{Set}$ for each $t \in T$, and an isomorphism

$$p \cong \sum_{t \in T} y^{p[t]}.$$

We call T the set of *p-types*, and for each type $t \in T$ we call $p[t]$ the set of *p-terms of type t*.

A *morphism* $\varphi\colon p \to p'$ of polynomial functors is simply a natural transformation between them. We denote the category of polynomial functors by **Poly**. It is called *cartesian* if for every map of sets $f\colon S \to S'$, the naturality square

$$\begin{array}{ccc} p(S) & \xrightarrow{p(f)} & p(S) \\ {\scriptstyle \varphi(S)}\downarrow & \lrcorner & \downarrow{\scriptstyle \varphi(S')} \\ p'(S') & \xrightarrow[p'(f)]{} & p'(S') \end{array}$$

is a pullback of sets. We denote the wide subcategory of polynomials and cartesian maps by $\mathbf{Poly}^{\mathbf{Cart}}$. ◊

For any polynomial $p = \sum_{t \in T} y^{p[t]}$, we have a canonical isomorphism $p(1) \cong T$; hence from now on we will denote p by

$$p = \sum_{I \in p(1)} y^{p[I]} \tag{3}$$

so that the p-types are written with upper-case letters, e.g. $I \in p(1)$, and its terms are written with corresponding lower-case letters, e.g. $i \in p[I]$.

Remark 2.2. Note that there is already some dependency in Definition 2.1; in order to define a polynomial we need a set I and then, *for each i in I*, we need a representable functor. Thus though we will use polynomial functors to explicate the syntax and rules of dependent types, but to do so relies on our meta-theory allowing us to already know what dependent types are. ◊

Remark 2.3. Using the Yoneda lemma, we can understand a morphism $p \to q$ in **Poly** to consist of two parts $(\varphi_1, \varphi^\sharp)$ as follows:

$$\varphi_1\colon p(1) \to q(1) \qquad \text{and} \qquad \varphi_I^\sharp\colon q[J] \to p[I], \tag{4}$$

where $J \coloneqq \varphi_1 I$.[7] That is, φ_1 is a function from p-types to q-types, and $\varphi_i^\sharp$ is a function on terms that *depends on a choice of position* $I \in p(1)$. We refer to φ_1 as the *on-types function* and to $\varphi^\sharp$ as the *backwards on-terms* function.

One can check that a map $\varphi : p \to q$ is cartesian iff the backwards-on-terms function $\varphi_I^\sharp$ is a bijection $p[I] \cong q[\varphi_1 I]$ for each type $I \in p(1)$. ◊

Definition 2.4 (Sum-product sets and list polynomials)**.** We say that a cardinal κ is a *sum-product cardinal* if it satisfies the following properties for any sets A and $(B_a)_{a \in A}$, whose disjoint union we denote by $S(A, B) \coloneqq \coprod_{a \in A} B_a$ and whose cartesian product we denote by $\Pi(A, B) \coloneqq \prod_{a \in A} B_a$:[8]

(i) κ is nonempty $0 < \kappa$;
(ii) κ is closed under its own sums: if $A < \kappa$ and each $B_a < \kappa$, then $S(A, B) < \kappa$;
(iii) κ is closed under its own products: if $A < \kappa$ and each $B_a < \kappa$, then $\Pi(A, B) < \kappa$;
(iv) there exists a set C_κ containing exactly one set of each cardinality $N < \kappa$.[9]

Given a sum-product cardinal κ, we define the *κ-list polynomial* to be

$$u_\kappa \coloneqq \sum_{N \in C_\kappa} y^N \qquad \text{◊}$$

Example 2.5. Given the axiom of choice, the size of any Grothendieck universe is a sum-product cardinal; indeed conditions (i) – (iii) are satisfied for any Grothendieck universe, and (iv) is satisfied assuming the axiom of choice.

There is only one finite cardinal with the above properties, namely $\kappa = 2$. Indeed, 0 fails condition (i), 1 fails condition (iii) because $\Pi(0, !) = 1 \not< 1$, and any $2 < \kappa < \mathbb{N}$ fails condition (ii) because by induction $\Pi(\kappa - 1, \kappa - 1) \not< \kappa$.

The easiest intuitive example to keep in mind for our paper is $\kappa = \mathbb{N}$,[10] so that $N < \mathbb{N}$ iff N is finite. The $\mathbb{N}$-list polynomial is

$$u_{\mathbb{N}} \coloneqq \sum_{N \in \mathbb{N}} y^N. \qquad \text{◊}$$

2.2 Composition and Dirichlet monoidal structures

Polynomial functors are closed under composition, which we denote by $- \triangleleft -$; the types and terms of $p \triangleleft q$ are given by the following formula:

$$\left(\sum_{I \in p(1)} y^{p[I]} \right) \triangleleft \left(\sum_{J \in q(1)} y^{q[J]} \right) \coloneqq \sum_{I \in p(1)} \sum_{J \colon p[I] \to q(1)} y^{\sum_{i \in p[I]} q[Ji]}. \tag{5}$$

[7] In this paper, we generally denote *function* application by juxtaposition, e.g. for $f : A \to B$ we write $fa \in B$ rather than $f(a)$ for the image of a under f. However, we denote *functor* application using parentheses, e.g. $p(1)$ to denote $p : \mathbf{Set} \to \mathbf{Set}$ applied to $1 = \{'1'\}$ in $\mathbf{Set}$.

[8] We write $S(A, B)$ rather than $\Sigma(A, B)$ because we already have used that symbol for a monad structure. Later we will define $\Sigma(A, B)$ to be this $S(A, B)$, so the distinction is not important.

[9] The set C_κ is structure, not property, but any two choices will yield isomorphic list polynomials, so we are not concerned with the distinction here.

[10] We write $\mathbb{N}$ rather than $\aleph$ for the cardinality of natural numbers.

This gives a (nonsymmetric) monoidal structure (**Poly**, y, $\triangleleft$). One can see that a type in $p \triangleleft q$ consists of a type $I \in p(1)$ and a type $(Ji) \in q(1)$ for every term $i \in p[I]$.[11] A term in it consists of a term $i \in p[I]$ and a term $j \in q[Ji]$.

We will also be interested in another monoidal product called *Dirichlet product* and denoted $- \otimes -$; the types and terms of $p \otimes q$ are given by the following formula:

$$\left(\sum_{I \in p(1)} y^{p[I]}\right) \otimes \left(\sum_{J \in q(1)} y^{q[J]}\right) := \sum_{(I,J) \in p(1) \times q(1)} y^{p[I] \times q[J]}. \tag{6}$$

This gives a symmetric monoidal structure (**Poly**, y, $\otimes$). A type in $p \otimes q$ is just a pair of types $(I, J) \in p(1) \times q(1)$ and a term of it is just a pair of terms $(i, j) \in p[I] \times q[J]$.

Proposition 2.6. *Both the $\triangleleft$ and the $\otimes$ products preserve cartesian maps.*

Proof. Suppose given cartesian maps $\alpha \colon p_1 \to p_2$ and $\beta \colon q_1 \to q_2$. Then $\alpha \triangleleft \beta$ is cartesian using the pullback definition Definition 2.1 and $\alpha \otimes \beta$ is cartesian using the bijection-on-directions criterion from Definition 2.3. □

Proposition 2.7. *The identity functor* **Poly** $\to$ **Poly** *can be regarded as a lax monoidal functor*

$$\mathit{Indep} \colon (\mathbf{Poly}, y, \triangleleft) \to (\mathbf{Poly}, y, \otimes), \tag{7}$$

Moreover the laxators (lax coherence maps), which we denote with the same symbol

$$\mathit{Indep} \colon (p \otimes q) \to (p \triangleleft q)$$

*are cartesian for, and natural in, $p, q \in$ **Poly**.*

Proof. Suppose given polynomials $p, q \in$ **Poly**; since *Indep* is merely a lax monoidal structure atop the identity functor, the only data in it is in its laxator, which we define using Definition 2.3 as follows.

On types, we send an index $(I, J) \in p(1) \times q(1)$ to $(I, J_!) \in p \triangleleft q(1)$, where $J_! \colon p[I] \to q(1)$ is constant J, sending each $i \in p[I]$ to $J \in q(1)$. Backwards on terms, we send $(i, j) \mapsto (i, j)$; indeed both Eqs. (5) and (6) have the same set of directions $\sum_{i \in p[I]} q[J] \cong p[I] \times q[J]$ when $J_! := J$ is independent of I. Thus it is clear that *Indep* is cartesian. We leave it to the reader to check the relevant axioms. □

Proposition 2.8. *There is a duoidality between $\otimes$ and $\triangleleft$, i.e. for every $p_1, p_2, q_1, q_2 \in$ **Poly** there is a cartesian map*

$$(p_1 \triangleleft p_2) \otimes (q_1 \triangleleft q_2) \xrightarrow{\mathit{Duoid}} (p_1 \otimes q_1) \triangleleft (p_2 \otimes q_2)$$

satisfying the usual properties.

[11] Elements of $p \triangleleft q$ will eventually be our dependent types. We will later write $I \prec J$ to denote (I, J) with $I \in p(1)$ and $(Ji) \in q(1)$ for each $i \in p[I]$. It is as though J branches on I.

Proof. On types we begin with $(I_1, I_2), (J_1, J_2) \in ((p_1 \triangleleft p_2) \otimes (q_1 \triangleleft q_2))(1)$, where $I_2 \colon I_1 \to p_2(1)$ and $J_2 \colon J_1 \to q_2(1)$. It is sent by $Duoid_1$ to $(I_1, J_1), (I_2', J_2') \in (p_1 \otimes q_1) \triangleleft (p_2 \otimes q_2)(1)$ where $I_2' \colon I_1 \times J_1 \to p_2(1)$ is given by first projecting onto I_1 and then applying I_2, and similarly $J_2' \colon I_1 \times J_1 \to q_2(1)$ is given by first projecting onto J_1 and then applying J_2.

Backwards on terms, we begin with $((i_1, j_1), (i_2, j_2)) \in (p_1 \otimes q_1) \triangleleft (p_2 \otimes q_2)[(I_1, J_1), (I_2', J_2')]$, where $i_2 \in I_2'(i_1, j_1) = I_2 i_1$ and $j_2 \in J_2'(i_1, j_1) = J_2 j_1$. It is sent by the isomorphism $Duoid^\sharp_{(I_1,I_2),(J_1,J_2)}$ to $((i_1, i_2), (j_1, j_2)) \in ((p_1 \triangleleft p_2) \otimes (q_1 \triangleleft q_2))[(I_1, I_2), (J_1, J_2)]$. We leave it to the reader to check the relevant axioms. □

Proposition 2.9 (κ-list monads)**.** *For any sum-product cardinal κ, there is a cartesian monad structure $(1, \Sigma)$ on the κ-list polynomial $u_\kappa \in \mathbf{Poly}^{\mathbf{Cart}}$.*

Proof. Recall that $u_\kappa = \sum_{N \in C_\kappa} y^N$, where C_κ is a choice of one set for every cardinality $N < \kappa$, and that whenever $A < \kappa$ and $B_a < \kappa$ for each set in $(B_a)_{a \in A}$, the set $S(A, B)$ isomorphic to their disjoint union also satisfies $S(A, B) < \kappa$.

The monoidal unit on u_κ corresponds to the cardinal $1 < \kappa$,[12] considered as a (cartesian) map $y \to u_\kappa$. The monoidal product $u_\kappa \triangleleft u_\kappa \to u_\kappa$ sends $A \in u_\kappa(1)$ and $B \colon u_\kappa[A] \to u_\kappa(1)$ to $S(A, B)$. The monad laws are satisfied because coproduct is a monoidal product and u_κ has exactly one position (element in C_κ) for each cardinality $N < \kappa$. □

Definition 2.10 (κ-list monads)**.** For any sum-product cardinal κ, we refer to the monad $(u_\kappa, 1, \Sigma)$ from Definition 2.9 as the *κ-list monad*. Note that lax monoidal functors, such as $Indep \colon (\mathbf{Poly}, y, \triangleleft) \to (\mathbf{Poly}, y, \otimes)$ from (7), send monoids to monoids. Hence, we also have a $\otimes$-monoid structure on u:

$$y \xrightarrow{1} u \qquad \text{and} \qquad u \otimes u \xrightarrow{Indep} u \triangleleft u \xrightarrow{\Sigma} u. \tag{8}$$

We denote this monoidal product simply using parens: $(-,-) \colon u \otimes u \to u$. ◊

After we define polynomial universes in the next section, we will show that for any κ as in Definition 2.5, the κ-list monad $(u_\kappa, 1, \Sigma)$ from Definition 2.9 is a universe.

3 List monads are polynomial universes

In Section 3.1, we define polynomial universes by a structure we call the *jump transformation* and a list of six axioms about it. These axioms provide the type and term constructors and the conversion rules we want for dependent type theory, as we will see in Section 4. We will show how to package them up in various categorically-pleasing ways, though none we know of so far captures the full strength of the axioms in Definition 3.1.[13] In Section 3.2 we will define a proposed jump transformation in the case of κ-list monads, and in Section 3.3 we show that our proposal works: the result is indeed a polynomial universe.

[12] Recall from Definition 2.5 that $1 = \Pi(0, !) < \kappa$.

[13] We would very much appreciate knowing of a neat categorical structure that houses the axioms in Definition 3.1.

3.1 Polynomial universe

Below is a definition for our main notion of interest—polynomial universes—that consists of some structures and properties. There may be a good category theoretic notion encompassing all these, but we have not found it. Rather than delay any longer, here it is.

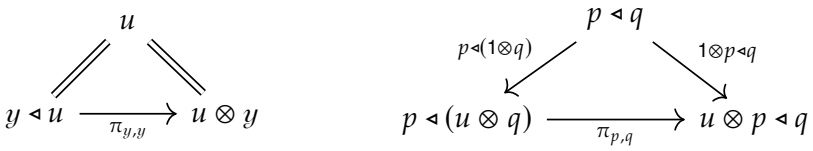
Definition 3.1. A *polynomial universe* consists of

1. a polynomial functor $u \in \mathbf{Poly}$
2. cartesian maps $1 \colon y \to u$ and $\Sigma \colon u \triangleleft u \to u$ forming a monad; and
3. a natural transformation between functors $\mathbf{Poly}^{\mathbf{Cart}} \times \mathbf{Poly} \to \mathbf{Poly}$ of the form[14]

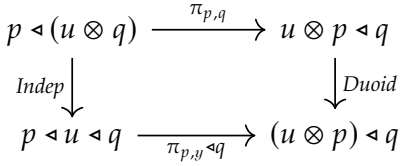
$$\pi_{p,q} \colon p \triangleleft (u \otimes q) \to u \otimes p \triangleleft q,$$

which we call the *jump transformation*;

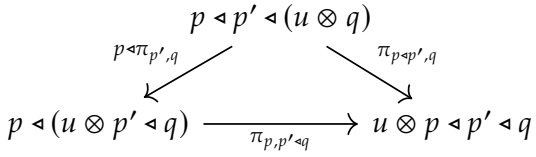
such that $\pi_{p,q}$ is natural in $p \in \mathbf{Poly}^{\mathbf{Cart}}$ and $q \in \mathbf{Poly}$, and makes the following diagrams commute:



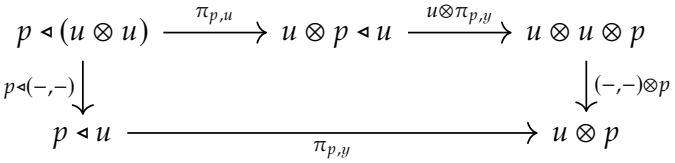
(9)



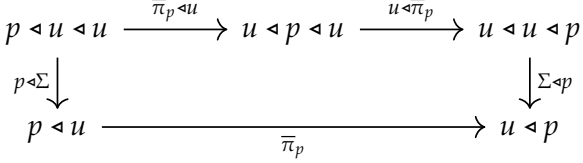
(10)



(11)



(12)



(13)

[14] We are sparing with our parentheses in this paper. We take the convention that $\triangleleft$ binds the tightest. Thus $u \otimes p \triangleleft q$ means $u \otimes (p \triangleleft q)$.

In the last diagram, $(-,-)\colon u \otimes u \to u$ is as in (8) and $\overline{\pi}_p\colon p \triangleleft u \to u \triangleleft p$ is the composite

$$p \triangleleft u \xrightarrow{\pi_p} u \otimes p \xrightarrow{Indep} u \triangleleft p. \qquad \lozenge \quad (14)$$

We refer to the map $\overline{\pi}\colon p \triangleleft u \to u \triangleleft p$ from (14), which is derived from the jump transformation, as the *associated jump map*.

Once we have a polynomial universe, we can package up various aspects of it in more familiar categorical terms, as we will see in **??** 3.2**??** 3.3. However neither of them captures the full strength of Definition 3.1.

Corollary 3.2. *Suppose that $(u, 1, \Sigma, \pi)$ is a polynomial universe. Then the following functors—labeled by where they send $p \in \mathbf{Poly}^{\mathbf{Cart}}$—as well as the natural transformation Indep between them, are monoidal:*

$$\left(\mathbf{Poly}^{\mathbf{Cart}}, y, \triangleleft\right) \quad \begin{array}{c} \xrightarrow{q \mapsto u \otimes p \triangleleft q} \\ \Downarrow Indep \\ \xrightarrow[q \mapsto u \triangleleft p \triangleleft q]{} \end{array} \quad \left(\mathbf{End}(\mathbf{Poly}), \mathrm{id}, \circ\right) . \qquad (15)$$

Proof. Suppose that u is a polynomial universe as in Definition 3.1, let $\overline{\pi}$ be the associated jump map (14), and let $\Phi, \overline{\Phi}\colon \mathbf{Poly}^{\mathbf{Cart}} \to \mathbf{End}(\mathbf{Poly})$ be the functors displayed in (15), i.e. $\Phi(p)(q) \coloneqq u \otimes p \triangleleft q$ and $\overline{\Phi}(p)(q) \coloneqq u \triangleleft p \triangleleft q$.

We first need to define the laxators $\phi, \overline{\phi}$ for Φ and $\overline{\Phi}$, and prove that the transformation *Indep* is monoidal. We thus need horizontal maps

$$\begin{array}{ccc} q & \xrightarrow{\phi_q} & u \otimes q \\ \| & & \downarrow Indep \\ q & \xrightarrow[\overline{\phi}_q]{} & u \triangleleft q \end{array} \qquad \begin{array}{ccc} u \otimes p_1 \triangleleft (u \otimes p_2 \triangleleft q) & \xrightarrow{\phi_{p_1,p_2,q}} & u \otimes p_1 \triangleleft p_2 \triangleleft q \\ {\scriptstyle Indep \triangleleft Indep}\downarrow & & \downarrow {\scriptstyle Indep} \\ u \triangleleft p_1 \triangleleft u \triangleleft p_2 \triangleleft q & \xrightarrow[\overline{\phi}_{p_1,p_2,q}]{} & u \triangleleft p_1 \triangleleft p_2 \triangleleft q \end{array}$$

natural in $p_1, p_2 \in \mathbf{Poly}^{\mathbf{Cart}}$ and $q \in \mathbf{Poly}$, that make those diagram commute. Define ϕ_q and $\overline{\phi}_q$ to respectively be the top and bottom maps in the commuting square:

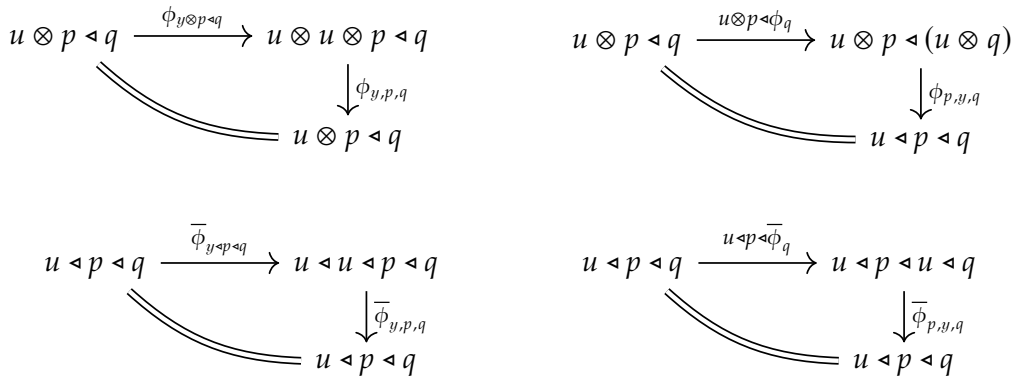
$$\begin{array}{ccc} q & \xrightarrow{1 \otimes q} & u \otimes q \\ \| & & \downarrow Indep \\ q & \xrightarrow[1 \triangleleft q]{} & u \triangleleft q \end{array}$$

Define $\phi_{p_1,p_2,q}$ and $\overline{\phi}_{p_1,p_2,q}$ to be the top and bottom composites in the rectangle

$$\begin{array}{ccccc} u \otimes p_1 \triangleleft (u \otimes p_2 \triangleleft q) & \xrightarrow{u \otimes \pi_{p_1, p_2 \triangleleft q}} & u \otimes u \otimes p_1 \triangleleft p_2 \triangleleft q & \xrightarrow{(-,-) \triangleleft p_1 \triangleleft p_2 \triangleleft q} & u \otimes p_1 \triangleleft p_2 \triangleleft q \\ {\scriptstyle Indep \triangleleft Indep}\downarrow & & \downarrow & & \downarrow {\scriptstyle Indep} \\ u \triangleleft p_1 \triangleleft u \triangleleft p_2 \triangleleft q & \xrightarrow[u \triangleleft \overline{\pi}_{p_1} \triangleleft p_2 \triangleleft q]{} & u \triangleleft u \triangleleft p_1 \triangleleft p_2 \triangleleft q & \xrightarrow[\Sigma \triangleleft p_1 \triangleleft p_2 \triangleleft q]{} & u \triangleleft p_1 \triangleleft p_2 \triangleleft q \end{array}$$

The left-hand square in this diagram commutes by (10) and the right-hand square commutes by (8) and the naturality of *Indep*.

It remains to show that the laxators ϕ and $\overline{\phi}$ satisfy the three commutative diagrams of a lax monoidal functor. The unit diagrams are:



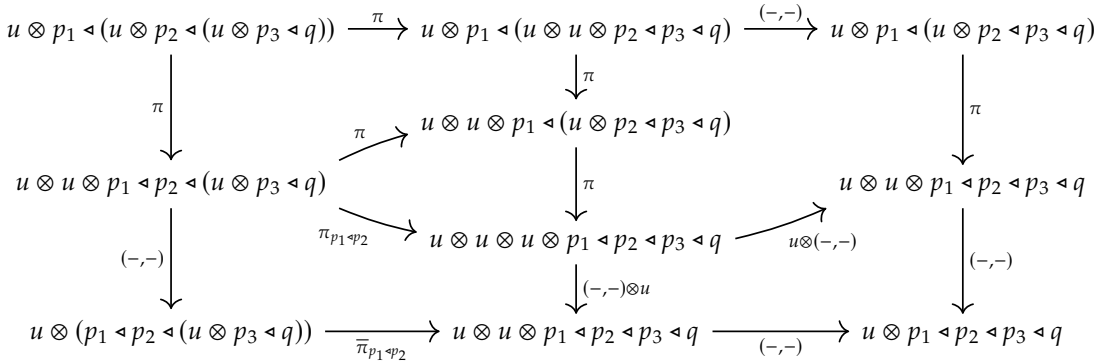
Both the top and bottom diagrams commute by the respective monad unit laws and the commutativity of the respective triangles in (9).

We conclude by showing that the associativity diagrams for Φ and $\overline{\Phi}$:

$$\begin{array}{ccc} u \otimes p_1 \triangleleft (u \otimes p_2 \triangleleft (u \otimes p_3 \triangleleft q)) & \xrightarrow{u \otimes p_1 \triangleleft \phi_{p_2,p_3,q}} & u \otimes p_1 \triangleleft (u \otimes p_2 \triangleleft p_3 \triangleleft q) \\ \downarrow{\scriptstyle \phi_{p_1,p_2,u \otimes p_3 \triangleleft q}} & & \downarrow{\scriptstyle \phi_{p_1,p_2 \triangleleft p_3,q}} \\ u \otimes p_1 \triangleleft p_2 \triangleleft (u \otimes p_3 \triangleleft q) & \xrightarrow[\phi_{p_1 \triangleleft p_2,p_3,q}]{} & u \otimes p_1 \triangleleft p_2 \triangleleft p_3 \triangleleft q \end{array} \tag{16}$$

$$\begin{array}{ccc} u \triangleleft p_1 \triangleleft u \triangleleft p_2 \triangleleft u \triangleleft p_3 \triangleleft q & \xrightarrow{u \triangleleft p_1 \triangleleft \overline{\phi}_{p_2,p_3,q}} & u \triangleleft p_1 \triangleleft u \triangleleft p_2 \triangleleft p_3 \triangleleft q \\ \downarrow{\scriptstyle \overline{\phi}_{p_1,p_2,u \triangleleft p_3 \triangleleft q}} & & \downarrow{\scriptstyle \overline{\phi}_{p_1,p_2 \triangleleft p_3,q}} \\ u \triangleleft p_1 \triangleleft p_2 \triangleleft u \triangleleft p_3 \triangleleft q & \xrightarrow[\overline{\phi}_{p_1 \triangleleft p_2,p_3,q}]{} & u \triangleleft p_1 \triangleleft p_2 \triangleleft p_3 \triangleleft q \end{array} \tag{17}$$

commute for any $p_1, p_2, p_3 \in \mathbf{Poly}^{\mathbf{Cart}}$ and $q \in \mathbf{Poly}$. Unpacking (16), we obtain the following diagram, where the four sides above agree with the four composite sides shown:[15]



The top-left and bottom-left squares commute by functoriality of monoidal products, and the bottom right square commutes by the monad associativity law. The rest is Definition 3.5: the little triangle commutes by (11) and the pentagon commutes by (12).

[15] We label the maps merely by "hints" for space reasons; e.g. the top maps labeled π and Σ are fully written as $u \triangleleft p_1 \triangleleft u \triangleleft p_2 \triangleleft \pi_{p_2} \triangleleft p_3 \triangleleft q$ and $u \triangleleft p_1 \triangleleft \Sigma \triangleleft p_2 \triangleleft p_3 \triangleleft q$.

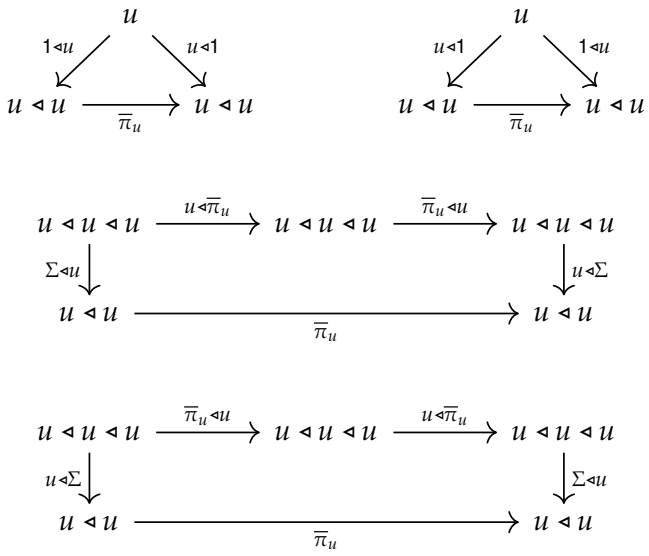
The commutativity of the other associativity diagram (17) is proved almost identically, except with four replacements made throughout: replace $\otimes$ by $\triangleleft$, replace $(-,-)$ by Σ, replace π by $\overline{\pi}$, and replace (12) with (13). □

Corollary 3.3 (Self distributive law)**.** *For a polynomial universe* $(u, 1, \Sigma, \pi)$*, the associated jump map*

$$\overline{\pi}_u : u \triangleleft u \longrightarrow u \triangleleft u$$

is a distributive law of the monad $(u, 1, \Sigma)$ *over itself.*

Proof. We need to prove that four diagrams commute:



The first two use the two triangles in (9); the first also uses naturality of $\overline{\pi}$ with respect to the cartesian map $1 : y \longrightarrow u$. The third uses Eq. (11) along with naturality of $\overline{\pi}$ with respect to the cartesian map $\Sigma : u \triangleleft u \longrightarrow u$. The fourth is simply (13). □

Distributive laws give rise to new monads. In this case, the new monad structure on $u \triangleleft u$ is roughly speaking a decategorified version of **Poly** inside of itself. For more on this, see this blog post.

Our next goal is to show that polynomial universes exist.

3.2 Defining a jump transformation

Definition 3.4 is the technical center of the paper, defining the jump transformation that will serve as the missing piece needed to define a polynomial universe, as in Definition 3.1. The idea is that a universe u should be able to absorb arbitrary dependencies into it, and in Definition 3.5 we show that the jump transformation from Definition 3.4 does the job.

The unexpected thing about the jump transformation is that to define it requires we use what we call *overflow*. It says that if you try to multiply too many things together, and the result is not a type in u, then we can define the result to be the empty type and all the desired equations still hold. One could instead restrict the domain (the

maximum size κ of types over which one is allowed to index such products), but then the statement becomes less attractive.[16] We also think that allowing the possibility of overflow is interesting, maybe even useful.

Definition 3.4 (Jump transformation π)**.** Let $u \coloneqq u_\kappa$ be a κ-list polynomial, and recall from Definition 2.4 that $\Pi(A, B) \in u(1)$ whenever $A < \kappa$ and $B_a < \kappa$ for each $a \in A$. Then for any $p \in \mathbf{Poly}^{\mathbf{Cart}}$ we define the *p-jump transformation*

$$\pi_p \colon p \triangleleft (u \otimes -) \to u \otimes p \triangleleft -$$

as follows.

Fix $q \in \mathbf{Poly}$. To give the q-component $\pi \coloneqq \pi_{p,q} \colon p \triangleleft (u \otimes q) \to u \otimes p \triangleleft q$ of the p-jump transformation, we follow Definition 2.3, first defining the function π_1 on types and then a function $\pi^\sharp$ backwards on terms.

A type in the family $p \triangleleft (u \otimes q)$ consists of a triple (I, A, J) where $I \in p(1)$, $A \colon p[I] \to u(1)$, and $J \colon p[I] \to q(1)$. We want to define a type in the family $u \otimes p \triangleleft q$, which consists of a triple (A', I', J') where $A' \in u(1)$, $I' \in p(1)$, and $J' \colon p[I'] \to q(1)$. We define $I' \coloneqq I$, $J' \coloneqq J$, and A' as follows:

$$A' \coloneqq \begin{cases} \Pi(p[I], A) & \text{if } \Pi(p[I], A) < \kappa \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

In a case of (I, A, J) for which the second condition is activated, we say that π *overflows*; otherwise we say π *succeeds*.

For the function $\pi^\sharp_{(I,A,J)} \colon (u \otimes p \triangleleft q)[(A', I, J)] \to (p \triangleleft (u \otimes q))[(I, A, J)]$ backwards on terms, there are two cases. If $A' = 0$ then $(u \otimes p \triangleleft q)[(A', I, J)] = 0$, so there is a unique map. Thus if π overflows one could say that the system "halts" in the sense that it has no terms, and any composite of other polynomials with it will also have no terms. Back to the primary case—i.e., if we assume that π succeeds—we need a map

$$\prod_{i \in p[I]} Ai \times \sum_{i \in p[I]} q[Ji] \to \sum_{i \in p[I]} (Ai) \times (Ji) \tag{19}$$

and we take it to be $(a, i, j) \mapsto (i, ai, ji)$.[17] ◊

Our goal in Section 3.3 is to show that this definition satisfies the properties necessary to define a polynomial universe.

Note that one can avoid the whole "overflow" issue by restricting the domain category throughout this paper to $\mathbf{Poly}^{\mathbf{Cart}}_{<\kappa} \subseteq \mathbf{Poly}^{\mathbf{Cart}}$, meaning that the jump map $\pi_{p,q} \colon p \triangleleft (u \triangleleft q) \to u \otimes p \triangleleft q$ would be restricted only to those p whose exponents have cardinality bounded by κ. We contemplated this move, but decided to allow arbitrary p in order to imagine dependencies on "real world", or "external" parameters.

[16] To make this restriction, we'd need to bake κ into Definition 3.1, making it explicit throughout and writing $\mathbf{Poly}^{\mathbf{Cart}}_\kappa$ in place of $\mathbf{Poly}^{\mathbf{Cart}}$.

[17] Note that the map $\pi^\sharp_{(I,A,J)}$ is not generally injective, because for a given input (a, i, j), most of the data in a is discarded; hence π is not cartesian. This has nothing to do with overflow, and hence could not be avoided by merely restricting $\mathbf{Poly}^{\mathbf{Cart}}$ to κ-bounded polynomials.

3.3 Main theorem

In this section we prove the main theorem of the paper, Definition 3.5.

Theorem 3.5 (Main theorem)**.** *Let* $(u, 1, \Sigma)$ *be a* κ*-list monad, and let* π *be the jump transformation from Definition 3.4. Then* $(u, 1, \Sigma, \pi)$ *forms a polynomial universe.*

Proof. The notion of polynomial universe was defined in Definition 3.1. We begin by noting that $\pi_{p,q} \colon p \triangleleft (u \otimes q) \to u \otimes p \triangleleft q$ is clearly natural in $q \in \mathbf{Poly}$. Although it would not be natural for arbitrary polynomial maps $\alpha \colon p \to p'$,[18] it is natural when α is cartesian. Indeed if $A \colon p[I] \to u(1)$ and $\alpha_1(I) = I'$ then the bijection $\alpha_I^\sharp$ extends to a bijection

$$\Pi(p[I], A) \cong \Pi(p'[I'], A')$$

where $A' = A \circ \alpha_I^\sharp \colon I' \to u(1)$, and π_p overflows iff $\pi_{p'}$ does. Since π is the y-component of π composed with *Indep*, it is natural in $p \in \mathbf{Poly}^{\mathbf{Cart}}$ as well.

We next consider the commutativity of the diagrams in (9). The map $y \triangleleft u \to u \otimes y$ sends $(1, A) \mapsto \Pi(1, A())$, i.e. the one-fold product of A with itself. This is isomorphic to A (and hence equal in C_κ). Thus the left hand triangle commutes. For the right-hand triangle, suppose given $I \in p(1)$. Under the left-hand composite map it is sent to $(\Pi(p[I], 1), I)$ and under the right-hand map it is sent to $(1, I)$; these are the same element of C_κ, since $\Pi(p[I], 1)$ is defined isomorphic to a product of 1's, which is isomorphic to 1. The maps on directions are also identical.

To prove that (10) commutes, suppose given $(I, A, J) \in p \triangleleft (u \otimes q)(1)$, i.e. $A \colon p[I] \to u(1)$ and $J \colon p[I] \to q(1)$. Around both composites it is sent to $((\Pi(I, A), I), J') \in (u \otimes p) \triangleleft q$ where J' is the composite $\Pi(I, A) \times I \to I \xrightarrow{J} q(1)$.

To prove that (11) commutes, it suffices to show for any set I, sets $(I'_i)_{i \in I}$, and sets $(A_{i,i'})_{i \in I, i' \in I'_i}$ with $A_{i,i'} < \kappa$, that first there is a bijection

$$\Pi(\Sigma(I, I'), A) \cong \Pi(I, \Pi(I', A)) \tag{20}$$

which there is, and that second the overflow handling agrees, as we now argue. If for any $i \in I$ there is an overflow $\Pi(I'_i, A_i) \geq \kappa$ then it will output 0, and the product of anything with 0 is 0; hence the two sides agree that the result is 0. Otherwise, if all $\Pi(I'_i, A_i)$ succeed, then if the mutual result (20) is too big, both sides again agree that the result is 0.

To prove that (12) commutes, suppose given (I, A, B) where $I \in p(1)$ and $A, B \colon p[I] \to u(1)$. It is easy to check that, for some $A_1, A_2 \in u(1)$, it is sent under the top composite

[18] Consider the unique map $\alpha \colon y^1 \to y^0$. The naturality diagram

$$\begin{array}{ccc} y^1 \triangleleft u & \xrightarrow{\pi_{y^1}} & u \otimes y^1 \\ {\scriptstyle \alpha \triangleleft u}\downarrow & & \downarrow{\scriptstyle u \otimes \alpha} \\ y^0 \triangleleft u & \xrightarrow[\pi_{y^0}]{} & u \otimes y^0 \end{array}$$

does not commute: starting with $(1, A)$, the top composite yields $(A, ())$, whereas the bottom composite yields $(0, ())$. Thus π, and hence $\overline{\pi}$ is not generally natural for noncartesian α.

map to (A_1, I) and under the bottom composite map it is sent to (A_2, I). Assuming neither overflows, these are

$$A_1 \cong \Pi(p[I], (A, B)) \qquad \text{and} \qquad A_2 \cong (\Pi(p[I], A), \Pi(p[I], B))$$

respectively, and these sets are isomorphic (hence equal in C_κ). The issue with overflows is a special case of that for (13), so we proceed to that case.

To prove that (13) commutes, suppose given $(I, A, B) \in p \triangleleft (u \triangleleft u)(1)$, where $A \colon p[I] \to u(1)$ and $B \colon \sum_{i:p[I]} Ai \to u(1)$. It is easy to check that, for some $A_1, A_2 \in u(1)$, it is sent under the top composite map to (A_1, I) and under the bottom composite map it is sent to (A_2, I). Assuming neither overflows, these are

$$A_1 \cong \Pi(p[I], \Sigma(A, B)) \qquad \text{and} \qquad A_2 \cong \Sigma(\Pi(p[I], A), \Pi(p[I], B))$$

respectively, and these sets are isomorphic (hence equal in C_κ). Here we did not check that the maps back on directions are equal, and they're not in general. This is why we need to move to univalent (HoTT) polynomials. So it remains to deal with the overflow situations.

Note that $\Pi(p[I], \Sigma(A, B)) \cong \Sigma(\Pi(p[I], A), \Pi(p[I], B))$ are bijective sets, so one overflows iff the other does. If they overflow then $\Pi(p[I], A \times B) \geq \kappa$ is too big, so $A_1 = 0$. In this case we also know that either $\Pi(p[I], A) \geq \kappa$ or $\Pi(p[I], Ba) \geq \kappa$ is too big, for some $a \in \Pi(p[I], A)$, by assumption on κ (see Definition 2.4). In the first case we will have $A_2 = \Sigma(0, \Pi(p[I], B))$ and in the second case we will have $A_2 = \Pi(p[I], \Sigma(A, 0))$; either way, the result is 0. Hence the diagram commutes, completing the proof. □

4 Consequences in the language of dependent types

Fix a polynomial universe $(u, 1, \Sigma, \pi)$. We now explain how the structures and axioms in Definition 3.1 manifest in the notation of dependent types.

4.1 Notation

Our notation is not meant to be authoritative; the mathematics of polynomial functors takes care of the meaning, so that we can be more relaxed about how we want to represent it. The following is only meant to have the level of formality of a math paper, not of a logic or philosophy paper.

Type collections, types, terms, and dependencies. We refer to a polynomial $p \in \mathbf{Poly}^{\mathbf{Cart}}$ as a *type collection*. We refer to each element $I, I' \in p(1)$ as a *type*, and we refer to each element $i \in p[I]$ as a *term of type* I and write $i : p[I]$.

We refer to u as the *type universe*; it is a type collection, but has more structure. Let $\mathsf{U} \coloneqq u(1)$ be the set of types in the universe polynomial u. We denote each element of U using an upper-case letter, A, B, etc., and call it a *base type*. Given a base type $A \in u(1)$, we denote $u[A]$ simply by $[A]$, since u is fixed. We denote each element of $[A]$ using the same letter in lower-case, $a, a' : [A]$, etc.; these are terms of type A.

We refer to y as the *unit* type collection. There is exactly one element of $y(1)$; we denote it 1. The set $y[1]$ has a unique element, and we denote it ().

An element of $p \triangleleft q(1) = (p \triangleleft q)(1)$ consists of an element $I : p(1)$ and an element $Ji : q(1)$ for each $i : p[I]$; we denote it by $(I \prec J) \in (p \triangleleft q)(1)$. Unlike standard notation, we want to be very clear when one type depends on another. Of course, the composite $r \coloneqq p \triangleleft q$ is itself just a polynomial, so $I \prec J$ is just a type (an element of $r(1)$); however if we receive $r = p \triangleleft q$ explicitly as a composite we may refer to $I \prec J$ as a *dependent type*: the q-type depends on a term of the p-type. A term of type $I \prec J$ consists of a pair (i, j) where $i : p[I]$ and $j : q[Ji]$; we denote it $(i, j) : (p \triangleleft q)[I \prec J]$.

Every time a type depends on terms of a previous type, we use a $\prec$-symbol; there may be several of these in a type definition. Our notation simply follows the polynomial formalism in the sense that whenever we see $\triangleleft$ in the polynomials, we see $\prec$ in the types. For example, a type in $p \triangleleft q \triangleleft r$ would be denoted $I \prec J \prec K$. Here we have for each $i \in p[I]$ a q-type $Ji \in q(1)$ and for each $j \in q[Ji]$ an r-type $Kij \in r(1)$. A term in $I \prec J \prec K$ is a triple (i, j, k) where $i \in p[I]$, $j \in q[Ji]$, and $k \in r[Kij]$.

An element of $(p \otimes q)(1)$ consists of a pair $(I, J) \in p(1) \times q(1)$; we call it a *product type*. It is explicitly independent in the sense that there is an isomorphism $p \otimes q \cong q \otimes p$. An element of $(p \otimes q)[(I, J)]$ consists of a pair $(i, j) \in p[I] \times q[J]$. However, the map $\mathit{Indep} \colon p \otimes q \to p \triangleleft q$ sends the independent pair (I, J) to the dependent type $(I \prec J) \coloneqq \mathit{Indep}(I, J)$.

Monoid structures on base types. The cartesian monoid structures $(u, 1, \Sigma)$ and $(u, 1, (-,-))$, and the map $\mathit{Indep} \colon u \otimes u \to u \triangleleft u$, are denoted as follows. The map $1 \colon y \to u$ provides a certain base type, which we again call $1 \in \mathsf{U}$. The map $\Sigma \colon u \triangleleft u \to u$ provides a function $u \triangleleft \mathsf{U} \to \mathsf{U}$, which we again call Σ. Thus for any dependent type $A \prec B : u \triangleleft \mathsf{U}$, we can form $\Sigma(A, B) : \mathsf{U}$. Similarly the map $(-,-) \colon u \otimes u \to u$ provides a function $\mathsf{U} \times \mathsf{U} \to \mathsf{U}$, which we again denote $(-,-)$. Thus for any pair of types $(A, B) : \mathsf{U} \times \mathsf{U}$ we can form $(A, B) : \mathsf{U}$.

$$\frac{}{1} \qquad \frac{A, B}{(A, B)} \qquad \frac{A \prec B}{\Sigma(A, B)}$$

The cartesianness of $1 \colon y \to u$ means that $1 : \mathsf{U}$ has a single term, which we again call $() : [1]$. The cartesianness of Σ means that a term of $\Sigma(A, B)$ consists of an element of $u \triangleleft u[(A \prec B)]$, i.e. a pair (a, b) where $a : [A]$ and $b : [Ba]$. Similarly, a term of (A, B) consists of an element $(a, b) : [A] \times [B]$. We can also write

$$\frac{=}{()} \qquad \frac{a, b}{\overline{\overline{(a, b)}}} \qquad \frac{a, b}{\overline{\overline{(a, b)}}} \tag{21}$$

The double lines mean that there is a bijection between terms at the top and terms at the bottom. Choosing a term in the empty context is the same as choosing a term of unit type. Choosing a term of independent A, B, meaning an $a : [A]$ and a $b : [B]$, is the same thing as choosing a term $(a, b) : [(A, B)]$ of the pair. And choosing a term of the dependent $A \prec B$, meaning a term $a : [A]$ and a term $b : [Ba]$, is the same thing as choosing a term $(a, b) : \Sigma(A, B)$.

The jump transformation. Recall that for any $p \in \mathbf{Poly}^{\mathbf{Cart}}$ and $q \in \mathbf{Poly}$, Definition 3.1 defines a notion of jump transformation

$$\pi_{p,q} : p \triangleleft (u \otimes q) \to u \otimes p \triangleleft q.$$

Thus, given a dependent type consisting of a type $I \in p(1)$ and, for each $i \in p[I]$ a base type $Ai : \mathsf{U}$ and a type $Ji \in q(1)$, we obtain a type denoted $(I \to A) : \mathsf{U}$, and we keep I, J as before. In symbols,

$$\frac{I \prec (A, J)}{(I \to A, I \prec J)}$$

Given terms $a : [I \to A]$, $i : p[I]$, and $j : q[Ji]$, we obtain a term $(i, ai, j) : p \triangleleft (u \otimes q)[I \prec (A, J)]$. We denote this backwards map on terms as follows:

$$\curvearrowleft \frac{i, (ai, j)}{(a, i, j)}$$

It means that starting with types $I \prec (A, J)$ and terms $a : [I \to A]$, $i : p[I]$, and $j : q[Ji]$, we obtain terms $i : p[I]$, $ai : [A]$ and $j : q[J]$. Unlike the case for 1, Σ, and $(-,-)$, this construction is not a bijection because $\pi_{u,u}$ is not cartesian, as explained in the footnote below Eq. (19).

4.2 Unpacking the results

Before we start unpacking the axioms from Definition 3.1, note that we already have the pairing laws from the fact that $(-,-)$ and Σ are cartesian monads.

$$\frac{A : \mathsf{U}}{(A, 1) \equiv (1, A)} \qquad \frac{A : \mathsf{U} \quad B : \mathsf{U} \quad C : \mathsf{U}}{(A, (B, C)) \equiv ((A, B), C)}$$

$$\frac{A : \mathsf{U}}{\Sigma(A, 1) \equiv \Sigma(1, A)} \qquad \frac{A \prec B \prec C : u \triangleleft u \triangleleft \mathsf{U}}{\Sigma(A, \Sigma(B, C)) \equiv \Sigma(\Sigma(A, B), C)}$$

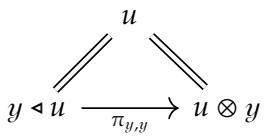
These also come with term conversions. The second pair looks exactly like the first pair because our notation for terms of product types is identical to that for terms of Σ types; hence we don't rewrite it:

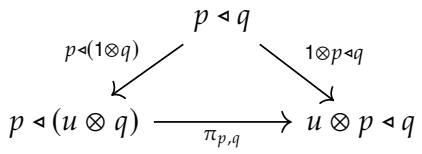
$$\frac{\quad a \quad}{\quad (a, ()) \equiv ((), a) \quad} \qquad \frac{\quad a, b, c \quad}{\quad (a, (b, c)) \equiv ((a, b), c) \quad}$$

The double lines mean that, since the monad operations are cartesian, the map backwards on terms is a bijection. Thus in both cases there is a one-to-one correspondence between terms below the line and above the line.

We now unpack the axioms from Definition 3.1; note that by Definition 3.5, these axioms are satisfied when u is a κ-list monad.

The first two diagrams





in (9) are translated tersely on top, or verbosely on bottom:

$$\frac{A}{(1 \to A) \equiv A} \qquad \frac{I \prec J}{(I \to 1) \equiv 1}$$

$$\frac{A}{(1 \to A, 1) \equiv (A, 1)} \qquad \frac{I \prec J}{((I \to 1), I \prec J) \equiv (1, I \prec J)}$$

From the verbose version we also get a term production rule:

$$\curvearrowleft \frac{(), a()}{a, ()} \qquad \curvearrowleft \frac{i, j}{(), i, j}$$

The rule provided by the diagram (10) is already inherent in the way our notation works (much like how, in string diagrams, associativity is there but invisible), so we do not write it out here.

The next diagram (11)

$$\begin{array}{ccc} & p \triangleleft p' \triangleleft (u \otimes q) & \\ {\scriptstyle p \triangleleft \pi_{p',q}} \swarrow & & \searrow {\scriptstyle \pi_{p \triangleleft p',q}} \\ p \triangleleft (u \otimes p' \triangleleft q) & \xrightarrow[\pi_{p,p' \triangleleft q}]{} & u \otimes p \triangleleft p' \triangleleft q \end{array}$$

is translated tersely as left, or verbosely as right[19]

$$\frac{I \prec I' \prec A}{((I \prec I') \to A) \equiv (I \to I' \to A)} \qquad \frac{I \prec I' \prec (A, J)}{((I \prec I') \to A, I \prec I' \prec J) \equiv (I \to I' \to A, I \prec I' \prec J)}$$

From the verbose version we also get a derivation rule backwards on terms:

$$\curvearrowleft \frac{i, i', (a i i', j)}{a, i, i', j}$$

The next diagram (12)

$$\begin{array}{ccccc} p \triangleleft (u \otimes u) & \xrightarrow{\pi_{p,u}} & u \otimes p \triangleleft u & \xrightarrow{u \otimes \pi_{p,y}} & u \otimes u \otimes p \\ {\scriptstyle p \triangleleft (-,-)} \downarrow & & & & \downarrow {\scriptstyle (-,-) \otimes p} \\ p \triangleleft u & & \xrightarrow[\pi_{p,y}]{} & & u \otimes p \end{array}$$

is translated tersely as left, or verbosely as right:

$$\frac{I \prec (A, B)}{I \to (A, B) \equiv (I \to A, I \to B)} \qquad \frac{I \prec (A, B)}{(I \to (A, B), I) \equiv ((I \to A, I \to B), I)}$$

[19] This syntax, which includes a $\prec$ symbol in the domain of an arrow $(I \prec I') \to A$, is unusual. But we can understand it in more classical notation below:

$$\prod_{i:I,\ i':I'i} Aii' \equiv \prod_{i:I} \prod_{i':I'i} Aii'$$

From the verbose version we also get a derivation rule backwards on terms:

$$\overleftarrow{\frac{i, ai, bi}{(a, b), i}}$$

The final diagram (13)

$$\begin{array}{ccccc} p \triangleleft u \triangleleft u & \xrightarrow{\overline{\pi}_p \triangleleft u} & u \triangleleft p \triangleleft u & \xrightarrow{u \triangleleft \overline{\pi}_p} & u \triangleleft u \triangleleft p \\ {\scriptstyle p \triangleleft \Sigma}\downarrow & & & & \downarrow{\scriptstyle \Sigma \triangleleft p} \\ p \triangleleft u & & \xrightarrow[\overline{\pi}_p]{} & & u \triangleleft p \end{array}$$

is translated tersely as left, or verbosely as right:

$$\frac{I \prec A \prec B}{I \longrightarrow \Sigma(A, B) \equiv \Sigma(I \longrightarrow A, I \longrightarrow B)} \qquad \frac{I \prec A \prec B}{(I \longrightarrow \Sigma(A, B)) \prec I \equiv \Sigma(I \longrightarrow A, I \longrightarrow B) \prec I}$$

From the verbose version we also get a derivation rule backwards on terms:

$$\overleftarrow{\frac{i, ai, bi}{(a, b), i}}$$

Again this looks exactly like the above case, but with Σ in place of $(-, -)$. This means that ambiguities in our notational system are appropriately washed away.

5 Future work

The most interesting remaining work is to find a clean category-theoretic definition for whatever is going on in Definition 3.1. There are a lot of moving pieces. Indeed, it begins with a duoidal category $(\mathbf{Poly}, y, \otimes, \triangleleft)$ equipped with a cartesian map $\mathit{Indep} \colon \otimes \longrightarrow \triangleleft$. In that already somewhat special context, we add a map $p \triangleleft (u \otimes q) \longrightarrow u \otimes (p \triangleleft q)$ with fairly expectable properties. But what is it all supposed to be? Is it a case of something category theorists already have a name for?

Another thing to do is to generalize the above considerations to polynomial universes in arbitrary copresheaf categories $\mathcal{C}$-**Set**. A first question is how to define polynomials in that context, because there are two reasonable generalizations. One is the usual one, which comes from maps $E \longrightarrow B$ in $\mathcal{C}$-**Set**; the other is that of parametric right adjoint functors $\mathcal{C}$-**Set** $\longrightarrow \mathcal{C}$-**Set**, which is strictly more general for nondiscrete $\mathcal{C}$.

Preliminary work suggests that **Poly** should be replaced by the category $\mathcal{C}$-**Set**$[\mathcal{C}]$ of parametric right adjoints $\mathcal{C}$-**Set** $\longrightarrow \mathcal{C}$-**Set**; see [Spi21]. In the case $\mathcal{C} = 1$, this yields 1-**Set**$[1] \cong$ **Poly**. It seems that $\mathbf{Poly}^{\mathbf{Cart}}$ should be replaced by the comma category, also known as the lax pullback, $(\mathcal{C} \downarrow \{\mathbf{Set}\})$ associated to the diagram of categories (not 2-categories)

$$C\text{-}\mathbf{Set} \xrightarrow{\mathit{elts}} \mathbf{Cat} \xleftarrow{\mathbf{Set}} \{*\}.$$

Here *elts* sends a copresheaf to its category of elements. In the case $\mathcal{C} = 1$, this lax pullback of $\mathbf{Set} \longrightarrow \mathbf{Cat} \xleftarrow{\mathbf{Set}} \{*\}$ yields $\mathbf{Poly}^{\mathbf{Cart}}$. Anyway, with these definitions at least

we know that the pieces we have used—including $\otimes, \triangleleft$, *Indep*, *Duoid*, etc.—all make sense; again see [Spi21] for relevant definitions.

References

[AN18] Steve Awodey and Clive Newstead. "Polynomial pseudomonads and dependent type theory". In: *arXiv* (2018). eprint: 1802.00997 (cit. on pp. 2, 4).

[Awo14] Steve Awodey. "Natural models of homotopy type theory". In: (2014). eprint: arXiv:1406.3219 (cit. on pp. 2, 3).

[GK12] Nicola Gambino and Joachim Kock. "Polynomial functors and polynomial monads". In: *Mathematical Proceedings of the Cambridge Philosophical Society* 154.1 (Sept. 2012), pp. 153–192 (cit. on p. 4).

[Mar75] Per Martin-Löf. "An Intuitionistic Theory of Types: Predicative Part". In: *Logic Colloquium '73*. Ed. by H.E. Rose and J.C. Shepherdson. Vol. 80. Studies in Logic and the Foundations of Mathematics. Elsevier, 1975, pp. 73–118 (cit. on p. 1).

[Spi21] David I. Spivak. *Functorial aggregation*. 2021. arXiv: 2111.10968 [math.CT] (cit. on pp. 19, 20).