HoTT Operads

***Abstract*—Internalising distinct classes of datatypes has been a longstanding pursuit in type theory. For example, the theory of containers captures strictly positive types, while combinatorial species capture finitely labelled structures. To date, however, there has been no similar attempt to internalise distinct classes of *operations* on datatypes. In this paper we show how the theory of operads, which extend species with a well-behaved notion of composition, provide a natural approach to internalising finitary operations. We present an internalisation of operads in homotopy type theory, which provides a generic framework for capturing and reasoning about operations with particular algebraic properties. All our results are formalised in Cubical Agda.**

I. Introduction

Formal theories of datatypes such as polynomial functors and strictly positive types have been extensively studied by type theorists. Containers were developed as an internal theory of datatypes with the form of a generalised polynomial [1]. The theory of containers captures both the strictly positive types [2] and ordinary polynomial functors. Similarly, combinatorial species have been used to capture finitely-labelled structures [3]. While containers represent datatypes by their 'shapes' and 'positions', combinatorial species describe the construction of a structure from a finite set of labels. One of the most prominent applications of both containers and species is generic programming in a dependent type theory. In this way, both theories give rise to an internalised calculus of datatypes.

With the existence of several internal theories of datatypes it appears natural to consider a similar calculus of *operations over datatypes*. However, perhaps surprisingly, this particular approach is much less explored in the type theory literature. The category theory literature provides a well-known tool for describing such algebraic structures. In particular, the notion of an *operad* is a generalisation of the standard concept of a category to a 'multicategory' (with one object), in which the source of a morphism is given by a finite sequence of objects. The theory of operads can be viewed as an extension to the theory of combinatorial species, in which an operad is simply a species together with a well-behaved notion of composition.

In this paper we present an internalised calculus of composable operations by realising the categorical notion of an operad in a suitable intensional type theory. In addition, we establish a formal relationship between combinatorial species and discretely-finite containers. More concretely, in terms of technical contribitions we:

- Prove that an internalisation of Bishop-finite sets is closed under dependent sums, which underpins the definition of a compositional structure over a (combinatorial) species;
- Introduce and internalise the concept of a pointed species (related to but distinct from the usual notion of pointing), which forms the underlying data of an operad;
- Internalise the definition of an operad, endomorphism operad and operad algebra, and demonstrate how the free operad can be constructed as a higher inductive family;
- Prove that every finitary container gives rise to a free construction of an operad, by establishing an isomorphism between species and discretely-finite containers;
- Prove that the type of algebras, over the free operad on a finitary container, is isomorphic to the type of algebras, restricted to h-sets, over that container.

The paper is aimed at programming language theorists who are interested in programming generically over classes of operations and category/type theorists who wish to reason about operads using a proof assistant. We assume a basic knowledge of type theory and category theory, but to make the paper more accessible we do not require expertise with containers, operads or homotopy type theory, and provide a brief introduction to each of these. All of our results are formalised in Cubical Agda, and are freely available online [4].

II. Background and Metatheory

To provide the necessary context for our work, we first give a brief introduction to the necessary background. We start with an introduction to the type theoretic concepts of a *container* and the related notion of a *discretely finite container*. We provide a standard categorical definition of *operads* and their associated category, and introduce our definition of *pointed species*, which we later use to characterise the underlying data of an operad in HoTT. Finally, we give a review of the HoTT literature and introduce notation used throughout this paper.

A. Containers

The theory of containers captures the notion of datatypes that can be formalised as a set of positions indexed over a set of shapes [2]. As an internalised theory of datatypes, containers give rise to generic programming in a suitable type theory.

In a type theory with a universe of types $\mathcal{U}$, closed under $\sum$ and $\prod$ type formers, a container $(S \rhd P)$ is given by a shape $S : \mathcal{U}$ and an S-indexed family of positions $P : S \to \mathcal{U}$. A container morphism between any two containers $(S \rhd P)$, $(T \rhd Q)$ is given by a map between the shapes $f : S \to T$, together with a (backwards) map between the positions $g : \prod_{(s:S)} Q\,(f\,(s)) \to P\,(s)$.

The container extension functor $[\![\_]\!] : \text{Container} \to \mathcal{U} \to \mathcal{U}$, is the functor whose action on a type $X : \mathcal{U}$ is given by

$$[\![S \rhd P]\!]\,(X) := \sum_{(s:S)} (P\,(s) \to X).$$

For any function $f : X \to Y$, the action of $[\![\_]\!]$ on f is given by $[\![S \rhd P]\!]\,(f, s, g) := (s, f \circ g)$

Discretely finite containers characterise the class of containers whose type of positions is 'finite' for every shape, e.g. the container $(\mathbb{N} \triangleright \mathrm{Fin})$ of finite lists. The standard definition of a *discretely finite object* can be given in any locally cartesian-closed category $\mathbb{C}$ with a natural numbers object $\mathbb{N}$.

Definition 1. *From [1], an object $B \in \mathbb{C}_{/A}$ is discretely finite iff there exists a morphism $u : A \to \mathbb{N}$, such that $B \cong u^{*}Fin$. Equivalently, for every global element $a : * \to A$, there exists a morphism $|B_a| : * \to \mathbb{N}$, such that the fibre of B over a is isomorphic to the finite cardinal $Fin_{|B_a|}$.*

Following Definition 1, we can translate the notion of a discretely finite or *finitary* container in a sufficient type theory as follows:

Definition 2. *A container $(S \triangleright P)$ is said to be discretely finite iff for every shape $s : S$ the corresponding type of positions $P(s)$ is discretely finite.*

We redefine the notion of a discretely finite container in HoTT, in Section IV (Definition 7), in which 'discretely finite object' is translated as 'Bishop-finite type'.

Discretely finite objects in a locally cartesian closed category are closed under finite product and sum constructions in the evident way. These constructions can be generalised to their dependent counterparts by (finitely) iterating addition and multiplication. A formal *constructive* proof of these closure properties, in HoTT, are provided in Section III and can also be found in the Cubical Agda formalisation of our work.

B. Operads

An operad is an interpretation of a combinatorial species as a collection of finitary operations with a well-behaved notion of composition. Given a monoidal category V, a (non-symmetric) V-operad is a V-species, $K : \mathbb{B} \to V$, extended with a family of composition maps

$$\circ_{n,ns} : K(n) \otimes K(ns_1) \otimes K(ns_2) \otimes ... \otimes K(ns_n) \to K(\Sigma\, ns),$$

for all $n \in \mathbb{B}$, $ns : [n] \to \mathbb{B}$, and a unit

$$e : * \to K(1),$$

subject to the evident unitality and associativity laws.

Every object of a monoidal category gives rise to a canonical operad known as the endomorphism operad. Given an object $X \in V$ the operations of the endomorphism operad over X are the morphisms

$$X^{\otimes n} \to X,$$

for every $n \in \mathbb{B}$. Composition maps are given by the evident plugging in of outputs into the corresponding inputs. The unit is given by the identity morphism on X.

Given operads $(K_1,\ \circ^1,\ e_1)$, $(K_2,\ \circ^2,\ e_2)$, an operad morphism is a family of maps

$$F_n : K_1(n) \to K_2(n),$$

for every $n \in \mathbb{B}$, such that

$$F_1 \circ e_1 = e_2,$$

$$(\circ^2) \circ (F_{ns_1} \otimes F_{ns_2} \otimes ... \otimes F_{ns_3}) = (F_{\Sigma ns}) \circ (\circ^1),$$

for every $ns : [n] \to \mathbb{B}$.

An algebra over a V-operad K is an object $X \in V$ together with an operad morphism from K to the endomorphism operad over X. This can be expressed as a family of maps

$$F : K(n) \to X^{\otimes n} \to X,$$

for all $n \in \mathbb{B}$. Intuitively, an operad algebra is a concrete realisation of an abstract collection of operations.

C. Pointed Species

Given a category V, a 'V-species' is a V-valued presheaf on the groupoid $\mathbb{B}$ of finite sets and bijections. The category of V-species, **VSpec**, is the category of V-valued presheafs over $\mathbb{B}$. If V has a terminal object, $*$, then for every $n \in \mathbb{N}$, there exists a category $\mathbf{VSpec}_{n+}$ whose objects are pairs consisting of a V-species

$$K : \mathbb{B}^{op} \to V,$$

and for every $X \in \mathbb{B}$, such that $|X| = n$, a global element

$$i_X : * \to K(X).$$

The morphisms between two pointed collections $(K_1,\ i),\ (K_2,\ j) \in \mathbf{VSpec}_{n+}$ are the natural families

$$\gamma_m : K_1(m) \to K_2(m),$$

satisfying

$$\gamma_n \circ i_X = j_X,$$

for all $X \in \mathbb{B}$. There is an evident forgetful functor

$$U : \mathbf{VSpec}_{n+} \to \mathbf{VSpec},$$

which forgets the global element of a pointed species.

Proposition 1. *If the category V has finite coproducts, the forgetful functor $U : \mathbf{VSpec}_{n+} \to \mathbf{VSpec}$ has a left adjoint.*

Proof. Given $K \in \mathbf{VSpec}$ and $X \in \mathbb{B}$, define

$$K_{n+}(m) = \begin{cases} * + K(m) & |m| = n \\ K(m) & \text{otherwise} \end{cases},$$

$$\eta_{K,m} = \begin{cases} \iota_{K(m)} & |m| = n \\ id_{K(m)} & \text{otherwise} \end{cases}.$$

To prove $-_{n+}$ is left adjoint to U it is sufficient to prove that for every $(K',\ i) \in \mathbf{VSpec}_{n+}$, and for every map $f_m : K(m) \to K'(m)$, there exists a unique map $g_m : K_{n+}(m) \to K'(m)$, such that $g_m \circ \eta_{K,m} = f_m$. Define

$$g_m = \begin{cases} [i_m,\ f_m] & |m| = n \\ f_m & \text{otherwise} \end{cases},$$

where $[i,\ f_m]$ is the copairing of i and f_m. Observe that $g_n \circ \iota_* = i$, and g is therefore a morphism of pointed species. This definition satisfies $g_m \circ \eta_{K,m} = f_m$ as required, and g_m is trivially the unique map satisfying this property. □

D. Homotopy Type Theory

We conclude our review of the background literature by giving a summary of the relevant ideas from homotopy type theory (HoTT). One of the fundamental concepts in HoTT is that of an identity type or 'path' type. In Martin-Löf type theory, the identity type $=_A$ over a type A is given as an inductive family with a single constructor $\mathbf{refl} : \prod_{(x:A)} x =_A x$. However, many models of HoTT, including many of the cubical variants, define 'identity' or 'path' in a different manner. Notably, cubical Agda is based on the CCHM model of cubical type theory.

The eliminator for the path type $=_A$, also known as the J-rule, states that for any type family $M : \prod_{(x,y:A)} x =_A y \to \mathcal{U}$ and term $t : \prod_{(x:A)} M(x, x, \mathbf{refl}_x)$, there exists a function

$$J(M,t) : \prod_{(x,y:A)} \prod_{(p:x=_A y)} M(x,y,p).$$

The J-eliminator comes equipped with the computation rule $J(M,t,x,x,\mathbf{refl}_x,u) := u$.

Given a type family $B : A \to \mathcal{U}$, and a path $p : x =_A y$ between terms $x, y : A$, the substitution along p is defined as $p^* := J(\lambda x.\lambda y.\lambda p.B(x) \to B(y), \lambda x.\lambda b.b, x, y, p)$. Given terms $b_1 : B(x)$ and $b_2 : B(y)$, and a path $p : x = y$, we abbreviate the path type $p^*(b_1) = b_2$ to $b_1 =^p b_2$. Function extensionality in HoTT is the principle which defines equality between functions pointwise. Concretely, a term of type $\prod_{(x:X)} f(x) =_X g(x)$, for two functions $f, g : X \to Y$, can be lifted to a term of type $f =_{X\to Y} g$.

Of particular importance in HoTT is the notion of 'homotopy level'. A type A is of homotopy level -2 or 'contractible' if there is $x : A$ such that for all $y : A$, the type $x =_A y$ is inhabited. Similarly, A is of homotopy level -1 or a 'proposition' if for all $x, y : A$, the type $x =_A y$ is inhabited. For all $n \geq 0$, we have that A is of homotopy level $n+1$. We write $\mathcal{U}^{\leq i}$ to denote the type $\sum_{(A:\mathcal{U})} \text{isOfHLevel}(i, A)$, where $\text{isOfHLevel}(i, A)$ witnesses that A is of homotopy level i.

Another important concept in HoTT is the univalence axiom. Univalence is often understood as the statement that 'equivalence is equivalent to equality'. An important consequence of this result is that the meta-theoretic statement that 'isomorphic types are internally indistinguishable' in standard Martin-Löf type theory, can be expressed internally in HoTT. Furthermore, since equivalence implies isomorphism, equivalent types can similarly be (internally) shown to be indistinguishable. In cubical models of type theory, univalence arises as a theorem, in contrast to its original formulation as an axiom [7].

Throughout this paper we shall make use of a single universe of types which we denote $\mathcal{U}$. Conceptually, a universe in Martin Löf type theory corresponds to a 'type of types' closed under a particular set of type formers, and we write $A : \mathcal{U}$ to mean a type A in universe $\mathcal{U}$. Universes naturally form a hierarchy $\mathcal{U}_1 < \mathcal{U}_2 < ...$, such that $\mathcal{U}_n : \mathcal{U}_{n+1}$, which can be used to avoid the inconsistent assumption $\mathcal{U} : \mathcal{U}$. To simplify the definitions and proofs in this paper we do not make explicit the (necessary) operations on universe levels, however, these are included in our Cubical Agda formalisation. We assume throughout that the universe $\mathcal{U}$ is closed under the usual constructions of dependent sums, products and W-types.

We conclude our review of HoTT by recalling the notion of 'higher inductive type'. Higher inductive types generalise the standard notion of an inductive type by adding path constructors, which define equations between the data constructors. An important example of a higher inductive type, which we make extensive use of in this paper, is propositional truncation. Given a type $X : \text{Type}$ its (-1)-truncation, denoted $\|X\|$, is given by the higher inductive type with the following constructors

$$|-| : X \to \|X\|$$
$$\mathbf{eq} : \prod_{(x,y:\|X\|)} x = y$$

Truncation elimination states that given a proposition $P : \mathcal{U}^{\leq -1}$, every map $f : X \to P$ gives rise to a map $\|f\| : \|X\| \to P$. The notion of truncation we consider in this paper comes equipped with the computation rule $\|f\|(|x|) := f(x)$.

III. Finite Sets

Recall that an operad is defined by a combinatorial species $K : \mathbb{B} \to \text{Set}$, with a compositional structure which respects unitality and associativity conditions. To internalise the notion of an operad, in HoTT, we must first consider an appropriate representation of $\mathbb{B}$, the 'groupoid of finite sets and bijections'. Any representation of $\mathbb{B}$ should aim to reflect its categorical structure, such as the cardinality of the hom-sets between its objects and its homotopy level (h-level).

Various notions of finiteness have previously been studied in the context of HoTT, such as Kuratowski-finiteness [8]. A type $X : \mathcal{U}$ is Kuratowski-finite if there exists a term $n : \mathbb{N}$ and a surjection $f : \text{Fin}_n \to X$, where Fin_n is the canonical finite set with n inhabitants. Our internalisation of operads makes use of a stronger notion of finiteness attributed to Bishop [9]. The relative 'strength' is induced by the condition that for X to be Bishop-finite, f must be a bijection. Equivalently, since Fin_n is an h-set, a type X is Bishop-finite if and only if there exists an isomorphism between X and Fin_n.

To internalise the notion of a Bishop-finite set in HoTT, we first recall the standard internal notion of an isomorphism. For any types $A, B : \mathcal{U}$, and functions $f : A \to B$, $g : B \to A$ we define $\text{section}_{f,g} := \prod_{(b:B)} f(g(b)) =_B b$ and $\text{retraction}_{f,g} := \prod_{(a:A)} g(f(a)) =_A a$. An isomorphism between A and B is then witnessed by a term of type

$$A \Leftrightarrow B := \sum_{(f:A\to B)} \sum_{(g:B\to A)} \text{section}(f,g) \times \text{retraction}(f,g).$$

Given a term $I : A \Leftrightarrow B$, we shall write $I^{\to} : A \to B$, $I^{\leftarrow} : B \to A$, $\text{left}_I : \text{section}_{I^{\to},I^{\leftarrow}}$, $\text{right}_I : \text{retraction}_{I^{\to},I^{\leftarrow}}$, for the corresponding projections. The inversion of an isomorphism $I : A \Leftrightarrow B$ will be denoted by $I^{-1} : B \Leftrightarrow A$. The product, coproduct and composition of isomorphisms are constructed pointwise in the obvious way. We denote each operation respectively by $\times$, $\uplus$ and $\circ$.

From this definition of isomorphism, we might naively internalise the type witnessing finiteness of $A : \mathcal{U}$ as $\sum_{(n:\mathbb{N})} A \Leftrightarrow \mathrm{Fin}_n$. Since Fin_n is an h-set, this is equivalent to the type $\sum_{(n:\mathbb{N})} A = \mathrm{Fin}_n$. However, it is well-known that given $B : \mathcal{U}$, a type $A : \mathcal{U}$ together with a path $p : A = B$ form a contractible pair [7]. Consequently, this definition is simply equivalent to the type $\mathbb{N}$ of natural numbers. In particular, this definition corresponds to Bishop-finite sets with an explicit total-ordering.

We construct a more appropriate representation of Bishop-finiteness by 'hiding' the isomorphism within a propositional truncation. Explicitly, we define finiteness in HoTT as

$$\mathrm{isFinite}\,(A) := \sum_{(n:\mathbb{N})} \|A \Leftrightarrow \mathrm{Fin}_n\|.$$

From this definition of finiteness, we can readily internalise the universe of Bishop-finite sets.

Definition 3. *A Bishop-finite set is a term of the type*

$$\mathit{FinSet} := \sum_{(X:\mathcal{U})} \mathit{isFinite}\,(X).$$

Given $A : \mathit{FinSet}$, we write

$$[\![A]\!] : \mathcal{U}, \quad |X| : \mathbb{N}, \quad A_{\cong} : \|A \Leftrightarrow \mathit{Fin}_{|A|}\|$$

for the respective projections of A. When constructing a proposition, the term constructed by elimination of $A_{\cong}$, of type $A \Leftrightarrow \mathit{Fin}_{|A|}$, will simply be denoted $A_{\cong}$.

As we might expect, every finite set A has a unique cardinality. Concretely, for all $m, n : \mathbb{N}$, and isomorphisms $I_m : A \Leftrightarrow \mathrm{Fin}_m$, $I_n : A \Leftrightarrow \mathrm{Fin}_n$, we can construct a path $p : m = n$. Note that we can eliminate the truncated isomorphisms only because the type of natural numbers $\mathbb{N}$ is an h-set, and thus $m = n$ is an h-prop. Since the second component of $\mathrm{isFinite}\,(A)$ is trivially an h-prop, we provide the equivalent proof that $\mathrm{isFinite}\,(A)$ is an h-prop.

Proposition 2. *The type $\mathit{isFinite}\,(A)$ is a proposition, i.e. for any*

$$(m_1,\ p_1),\ (m_2,\ p_2) : \sum_{(n:\mathbb{N})} \|A \Leftrightarrow \mathit{Fin}_n\|,$$

there exist terms $q : m_1 = m_2$ and $r : p_1 =^q p_2$.

Proof. A term of type $p_1 = p_2$ is given by the path constructor of a propositional truncation. Recall that the type $\mathbb{N}$ of natural numbers is an h-set, thus the type $m = n$ is a proposition. Accordingly, to construct a term of type $m_1 = m_2$, we can use the eliminator of a propositional truncation. Importantly, we are able to use the underlying isomorphisms of p_1 and p_2 to construct $m_1 = m_2$. Denote these isomorphisms by $\hat{p_1} : A \Leftrightarrow \mathrm{Fin}_{m_1}$ and $\hat{p_2} : A \Leftrightarrow \mathrm{Fin}_{m_2}$ respectively. We can construct the composite isomorphism $\hat{p_2} \circ \hat{p_1}^{-1} : \mathrm{Fin}_{m_1} \Leftrightarrow \mathrm{Fin}_{m_2}$. Finally, it is necessary to show

$$\mathrm{Fin}_{m_1} \Leftrightarrow \mathrm{Fin}_{m_2} \to m_1 = m_2.$$

Of course, this is a proof that the type family Fin is injective. Perhaps surprisingly, this proof is more involved than one might first expect. While we don't prove the proof here, a sketch of this proof is described in [3], and is included in the (standard) library of cubical Agda. □

Next, we study the homotopy level of Bishop-finite sets and the corresponding universe FinSet. In particular, we justify the nomenclature of Bishop-finite *set* by proving that every $A : \mathrm{FinSet}$ is indeed an h-set. Recalling that our intended outcome is to reflect the structure of the *groupoid* of finite sets and bijections, we also prove that the type FinSet is an h-groupoid.

Proposition 3. *For any $A : \mathit{FinSet}$, $[\![A]\!] : \mathcal{U}$ is an h-set. Equivalently, for any $x, y : [\![A]\!]$ the type $x = y$ is a proposition.*

Proof. For any $n : \mathbb{N}$, the type Fin_n is an h-set. Since homotopy level is respected by isomorphism we simply transport the proof $p : \prod_{(i,j:\mathrm{Fin}_n)} \mathrm{isProp}(i = j)$ along the isomorphism $X_{\cong}$. We observe that we can access the underlying isomorphism $X_{\cong}$, since for any $A : \mathrm{FinSet}$, the type $\prod_{(x,y:A)} \mathrm{isProp}\,(x =_A y)$ of proofs that A is an h-set, is itself a proposition. □

It is a well-known theorem in HoTT that equivalences preserve homotopy level. Consequently, to show that the universe FinSet is an h-groupoid, it is sufficient to show that the path type $[\![A]\!] = [\![B]\!]$ is an h-set and establish an equivalence $([\![A]\!] = [\![B]\!]) \simeq (A = B)$.

Lemma 1. *Given finite sets $A, A' : \mathit{FinSet}$, the type of paths between them, $A = A'$, is equivalent to the type of paths between their underlying sets $[\![A]\!] = [\![A']\!]$.*

Proof. While we do not provide the construction here, it can be shown that for any $X : \mathcal{U}$, family of propositions $B : Y \to \mathcal{U}^{\leq -1}$, and terms $x, y : \sum_X Y$, congruence applied to the first projection map, i.e. $\mathrm{cong}\,(\pi_1) : x = y \to \pi_1\,(x) = \pi_1\,(y)$, is an equivalence. Since the type $\mathrm{isFinite}\,(A)$ is a proposition for any $A : \mathrm{FinSet}$, the desired construction is given by inverting the equivalence $\mathrm{cong}\,(\pi_1)$. □

Theorem 4. *The type FinSet of finite sets is an h-groupoid.*

Proof. It is a known theorem in HoTT that for any n-types $X, Y : \mathcal{U}$, the path type $X = Y$ is an n-type. Since equivalences preserve homotopy level, and by Proposition 3 the type $[\![A]\!]$ is an h-set for every $A : \mathrm{FinSet}$, the equivalence established in Lemma 1 can be used to prove that the path type $A = B$ is an h-set for all $A, B : \mathrm{FinSet}$. □

We will now proceed to prove that Bishop-finite sets are closed under a collection of important type formers, such as coproduct and dependent sum. In particular, we show that to prove the universe FinSet is closed under a dependent type former, it is sufficient to prove closure of $\mathrm{Fin} : \mathbb{N} \to \mathcal{U}$ under that construction. First, we introduce a general definition for the closure of a family of types under a dependent type former.

Definition 4. *Given a type $A : \mathcal{U}$ and an A-indexed family $B : A \to \mathcal{U}$, we say the family B is closed under a dependent*

type construction $\Psi : \prod_{(A:\mathcal{U})} (A \to \mathcal{U}) \to \mathcal{U}$*, if there exists an operation*

$$\_ \cdot \_ : \prod_{(a:A)} (B\,(a) \to A) \to A$$

such that for all $a : A$*,* $as : B\,(a) \to A$*, there exists an isomorphism* $\Psi\,(B\,(a), B \circ as) \Leftrightarrow B\,(a \cdot as)$*, i.e. there exists a term of type*

$$\prod_{(a:A)} \prod_{(as:B(a)\to A)} \Psi\,(B\,(a), \lambda b.B\,(as\,(b))) \Leftrightarrow B\,(a \cdot as)\,.$$

We will additionally require the following lemma which states that a dependent pair of isomorphisms can be lifted, by function congruence, over a (dependent) family of types.

Lemma 2. *Given types* $A, A' : \mathcal{U}$*, together with type families* $B : A \to \mathcal{U}$*,* $B' : A' \to \mathcal{U}$*, if there exists an isomorphism* $I : A \Leftrightarrow A'$*, and a family of isomorphisms* $J : \prod_{(a:A')} B\,(I^{\leftarrow}\,(a)) \Leftrightarrow B'\,(a)$*, then for every type family* $\psi : \prod_{(X:\mathcal{U})} (X \to U) \to U$*, there exists an isomorphism* $\psi_{\Leftrightarrow} : \psi\,(A, B) \Leftrightarrow \psi\,(A', B')$*.*

Proof. We provide a sketch of the proof here, and a more detailed construction is given in our cubical Agda formalisation. By application of univalence to I and J, we can construct a path $p : A = A'$, and a family of paths $q : \prod_{(a:A')} B\,(p^{\leftarrow}\,(a)) = B'\,(a)$. By function congruence, and the definition of path spaces in $\sum$-types, there is a path $r : \psi\,(A, B) = \psi\,(A', B')$. We construct the desired isomorphism by a second application of univalence. □

We can now state, and prove, the following theorem on the closure properties of Bishop-finite sets:

Theorem 5. *To prove that Bishop-finite sets are closed under a dependent type construction* $\Psi : \prod_{(A:\mathcal{U})} (A \to \mathcal{U}) \to \mathcal{U}$*, it is sufficient to prove that the family of finite cardinals* $\mathit{Fin} : \mathbb{N} \to \mathcal{U}$ *is closed under* Ψ*.*

Proof. By definition of closure of Fin under Ψ, there exists an operation $\_ \cdot \_ : \prod_{(n:\mathbb{N})} (\mathrm{Fin}_n \to \mathbb{N}) \to \mathbb{N}$, and a family of isomorphisms

$$\mathbf{I}_{\_,\_} : \prod_{(n:\mathbb{N})} \prod_{(ns:\mathrm{Fin}_n \to \mathbb{N})} \Psi\left(\mathrm{Fin}_n, \lambda i.\mathrm{Fin}_{ns(i)}\right) \Leftrightarrow \mathrm{Fin}_{(n \cdot ns)}.$$

To prove closure of Bishop-finite sets under Ψ, it is necessary and sufficient to show that for all $A : \mathrm{FinSet}$ and $B : [\![A]\!] \to \mathrm{FinSet}$, the type $\Psi_{A,B} := \Psi\,([\![A]\!], \lambda a.[\![B\,(a)]\!])$ is Bishop-finite. We first recall that the type $\mathrm{isFinite}\,(\Psi_{A,B}) : \mathcal{U}$, is a proposition (Proposition 2). Accordingly, by elimination of the respective (-1)-truncated isomorphisms, we have an explicit isomorphism $\iota_A : [\![A]\!] \Leftrightarrow \mathrm{Fin}_{|A|}$, and by application of *finite choice*, a family of isomorphisms $\iota_B : \prod_{(a:A)} [\![B\,(a)]\!] \Leftrightarrow \mathrm{Fin}_{|B(a)|}$. The cardinality of the Bishop-finite set $\Psi_{A,B} : \mathcal{U}$ is given by the term $|A| \cdot |B| : \mathbb{N}$, where $|B| : \mathrm{Fin}_{|A|} \to \mathbb{N}$ is the term $\lambda i.\,|B\,(\iota_A^{\leftarrow}\,(i))|$. The final step of the proof requires construction of an isomorphism $p : \psi_{A,B} \Leftrightarrow \mathrm{Fin}_{|\psi_{A,B}|}$, witnessing the finiteness of the type $\psi_{A,B}$. By post-composition with the isomorphism $\mathbf{I}_{|A|,|B|} : \psi\left(\mathrm{Fin}_{|A|}, \lambda i.\mathrm{Fin}_{|B|(i)}\right) \Leftrightarrow \mathrm{Fin}_{|A|\cdot|B|}$, it suffices to construct an isomorphism $\psi_{\Leftrightarrow} : \psi_{A,B} \Leftrightarrow \psi\left(\mathrm{Fin}_{|A|}, \lambda i.\mathrm{Fin}_{|B|(i)}\right)$, which follows from Lemma 2. □

Before continuing with the proofs of explicit closure properties of the universe FinSet, we first introduce the following lemma which asserts that if a family of types B is closed under a binary type construction, then given a distinguished unit element, B is closed under any finite iteration of that construction. In particular, we will use this to prove closure of Fin under dependent sum by (finitely) iterating the coproduct construction.

Lemma 3. *Given a type* $A : \mathcal{U}$*, family* $B : A \to \mathcal{U}$*, together with binary operations* $+ : A \to A \to A$*,* $\_ \cdot \_ : \mathcal{U} \to \mathcal{U} \to \mathcal{U}$*, an element* $\varepsilon : A$*, and a family of isomorphisms* $\mathbf{I} : \prod_{(a,a':A)} B\,(a) \cdot B\,(a') \Leftrightarrow B\,(a + a')$*, then for every* $n : \mathbb{N}$*,* $as : \mathit{Fin}_n \to \mathbb{N}$*, there is an isomorphism* $\mathit{fold}_n\,(\_ \cdot \_, B\,(\varepsilon), \lambda i.B\,(as\,(i))) \Leftrightarrow B\,(\mathit{fold}_n\,(+, \varepsilon, as))$*. We note that* fold_n *is the well-known function which starts with a given initial element and folds a binary operation over* n *elements.*

Proof. Proof follows by induction on n, and we direct readers to our cubical Agda formalisation for a full proof of this lemma. We note that our construction requires the univalence principle to prove that a given function preserves isomorphisms. □

We now prove the first important closure property of the universe of Bishop-finite sets, namely FinSet is closed under (binary) coproducts.

Proposition 6. *The type FinSet is closed under binary coproducts, i.e. given* $X,\ Y : \mathit{FinSet}$*, there exist terms* $k : \mathbb{N}$ *and* $p : [\![X]\!] \uplus [\![Y]\!] \Leftrightarrow \mathit{Fin}_k$*.*

Proof. Given any $m, n \in \mathbb{N}$, we start by constructing a term

$$q_{m,n} : \mathrm{Fin}_m \uplus \mathrm{Fin}_n \Leftrightarrow \mathrm{Fin}_{m+n},$$

which witnesses the isomorphism between $\mathrm{Fin}_m \uplus \mathrm{Fin}_n$ and Fin_{m+n}. The map $q^{\rightarrow}_{m,n} : \mathrm{Fin}_m \uplus \mathrm{Fin}_n \to \mathrm{Fin}_{m+n}$ is given by the co-pairing of the two injections

$$\iota_m : \mathrm{Fin}_m \hookrightarrow \mathrm{Fin}_{m+n},\ \iota_n : \mathrm{Fin}_n \hookrightarrow \mathrm{Fin}_{m+n},$$

where ι_m maps each $x : \mathrm{Fin}_m$ to $x : \mathrm{Fin}_{m+n}$, and ι_n maps each $i : \mathrm{Fin}_n$ to $m + i : \mathrm{Fin}_{m+n}$. The inverse map $q^{\leftarrow}_{m,n} : \mathrm{Fin}_{m+n} \to \mathrm{Fin}_m \uplus \mathrm{Fin}_n$ is defined inductively by $q_{0,n}\,(i) := \mathbf{inr}\,(i)$, $q_{1+m,n}\,(0) := \mathbf{inl}\,(0)$, $q_{1+m,n}\,(1 + i) := [\ \mathbf{suc}, \mathit{id}\ ]\,(q_{m,n}\,(i))$.

We assert, without construction, the existence of the evident (but verbose) proof terms witnessing that $q^{\rightarrow}_{m,n}$ and $q^{\leftarrow}_{m,n}$ are indeed isomorphic. The construction of the term $p : [\![X]\!] \uplus [\![Y]\!] \Leftrightarrow \mathrm{Fin}_{|X|+|Y|}$ is given by $q_{|X|,|Y|} \circ (X_{\cong} \uplus Y_{\cong})$. □

The proof that FinSet is closed under binary coproducts can be used to show the key result that Bishop-finite sets are closed under the dependent sum construction.

Proposition 7. *The type FinSet is closed under* $\sum$*, i.e. given* $X : \mathit{FinSet}$*,* $Y : [\![X]\!] \to \mathit{FinSet}$*, there exist terms* $k : \mathbb{N}$ *and* $p : \sum_{x:[\![X]\!]} [\![Y(x)]\!] \Leftrightarrow \mathit{Fin}_k$*.*

Proof. By Theorem 5, it is sufficient to show that the family Fin is closed under dependent sum, i.e., given $n : \mathbb{N}$ and $ns : \mathsf{Fin}_n \to \mathbb{N}$, it is necessary to show there exists a $k : \mathbb{N}$, such that there is an isomorphism $\mathbf{I} : \left(\sum_{(i:\mathsf{Fin}_n)} \mathsf{Fin}_{ns(i)}\right) \Leftrightarrow \mathsf{Fin}_k$. We define k to be the finite summation over the natural numbers ns, i.e. by application of the inductive function

$$\Sigma_0\ ns := 0, \qquad \Sigma_{\mathsf{suc}(n)}\ ns := ns\,(0) + \Sigma_n\ \lambda i.ns\,(\mathsf{fsuc}\,(i))\,.$$

We similarly define an intermediate family of types

$$\biguplus_{0} ns := \mathsf{Fin}_0 \qquad \biguplus_{\mathsf{suc}(n)}\ ns := \mathsf{Fin}_{ns(0)} \uplus \biguplus_{n} (\lambda i.ns\,(\mathsf{fsuc}\,(i)))\,.$$

By Lemma 3 and Proposition 6, there exists an isomorphism $\mathbf{p} : \biguplus_n ns \Leftrightarrow \mathsf{Fin}_{\Sigma_n ns}$. To conclude this proof, it suffices to construct a much simpler isomorphism $\mathbf{q} : \left(\sum_{(i:\mathsf{Fin}_n)} \mathsf{Fin}_{ns(i)}\right) \Leftrightarrow \biguplus_n ns$. We direct interested readers to our cubical Agda formalisation for a precise construction of q. □

We conclude our proofs on the closure properties of Bishop-finite sets here. However, we guide interested readers to our cubical Agda formalisation for the proofs that FinSet is also closed under binary products, dependent product, and even isomorphism, equivalence and equality constructions.

Proposition 8. *The type of finite sets of cardinality* 1 *is contractible, with* $\top$: *FinSet as its centre of contraction.*

Proof. It is evident and well-known that the type of contractible types is itself contractible, with $\top : \mathcal{U}$ as its centre of contraction. By Lemma 1, given any finite set A, a path $[\![A]\!] = \top$ can be lifted to $A = \top$. Therefore, it is sufficient to simply prove that any finite set of cardinality 1 is contractible. We begin by observing the well-known theorem that the type witnessing contractibility, i.e. $\mathsf{isContr}\,(X) := \sum_{(x:X)} \prod_{(y:X)} x = y$, is a proposition. Accordingly, by use of the propositional truncation elimination rule, we may access the explicit isomorphism between $[\![A]\!]$ and $\mathsf{Fin}_{|A|}$. By definition of A having cardinality 1, there is a path $p : |A| = 1$. By transport over p, and by application of univalence we can construct a path between $[\![A]\!]$ and Fin_1. Since there is a trivial path between Fin_1 and $\top$, a path of type $[\![A]\!] =_{\mathcal{U}} \top$ follows simply by path composition. □

Proposition 9. *Given two finite sets* X, Y : *FinSet, a truncated path,* $\|X = Y\|$*, can be constructed from a path* $p : |X| = |Y|$ *between their cardinalities.*

Proof. We immediately observe that the truncated type $\|X = Y\|$ is trivially a proposition. Accordingly, the isomorphisms $X_{\cong} : [\![X]\!] \Leftrightarrow \mathsf{Fin}_{|X|}$, $Y_{\cong} : [\![Y]\!] \Leftrightarrow \mathsf{Fin}_{|Y|}$, may be used in the construction of a term of type $\|X = Y\|$. By Lemma 1, a path $X = Y$ is constructible from a path $[\![X]\!] = [\![Y]\!]$. To construct a path $[\![X]\!] = [\![Y]\!]$, we first construct an isomorphism $I : [\![X]\!] \Leftrightarrow \mathsf{Fin}_{|Y|}$ by substitution along the path p over the term $X_{\cong}$. Composition of I with the inversion of $Y_{\cong}$ yields an isomorphism $[\![X]\!] \Leftrightarrow [\![Y]\!]$. By application of univalence we obtain the desired path $[\![X]\!] = [\![Y]\!]$. □

IV. Species

In contrast to the necessarily involved treatment of finite sets, combinatorial species enjoy a much simpler internalisation in HoTT. Following from the internalisation of Bishop-finite sets, a combinatorial species is any family $K : \mathsf{FinSet} \to \mathcal{U}$. The justification for this definition follows from congruence of functions, i.e., paths are respected by functions. By application of univalence, functions must similarly respect isomorphism, and in this way K is functorial with respect to isomorphisms between finite sets. We can similarly define a combinatorial species whose codomain is restricted to i-types, as a family of the form $\mathsf{FinSet} \to \mathcal{U}^{\leq i}$. A morphism between two species $K_1, K_2 : \mathsf{FinSet} \to \mathcal{U}^{\leq i}$, is a natural family $\eta : \prod_{(A:\mathsf{FinSet})} K_1\,(A) \to K_2\,(A)$, and we denote the type of morphisms between K_1 and K_2 as $K_1 \Rightarrow K_2$.

Definition 5. *Given any* n*, the type of pointed* n*-species (of homotopy level* i*) is given by*

$$*_n\text{-}\mathit{Species}_i := \prod_{(A:\mathit{FinSet})} \sum_{(X:\mathcal{U}^{\leq i})} (n = |A|) \to X.$$

To simplify notation, for $K : *_n\text{-Species}_i$, $A : \mathsf{FinSet}$, we will often write $K\,(A)$ in place of the term $\pi_1\,(K\,(A)) : \mathcal{U}^{\leq i}$. In a similar fashion, for $p : n = |A|$, we denote the term $\pi_2\,(K\,(A))\,(p) : K\,(A)$ by $\tilde{K}_A\,(p)$.

We recall that a morphism between two pointed n-species is a natural family of point-preserving maps between their underlying species. In HoTT, we can witness preservation of the point by a path. However, given a pointed species morphism with codomain $K : *_n\text{-Species}_i$, for $i \geq 1$, a naive internalisation inadvertently adds proof relevance to the preservation of the point. We therefore enforce proof-irrelevance by propositionally truncating the relevant path.

Definition 6. *The hom between two pointed* n*-species* $K : *_n\text{-}\mathit{Species}_i$, $K' : *_n\text{-}\mathit{Species}_j$ *is given by the type*

$$K \Rightarrow_* K' := \prod_{(A:\mathit{FinSet})} \sum_{(f:K(A)\to K'(A))} \prod_{(p:n=|A|)} \left\| f\left(\tilde{K}_A\,(p)\right) = \tilde{K'}_A\,(p) \right\|.$$

Given $f : K \Rightarrow_* K'$, $A : \mathsf{FinSet}$ we often write $f_A\,(k)$, for $k : K\,(A)$, to denote the term $\pi_1\,(f\,(A))\,(k) : K'\,(A)$. Similarly, for $p : n = |A|$, we write $\tilde{f}_A\,(p)$ to denote the term $\pi_2\,(f\,(A))\,(p) : \left\| f_A\left(\tilde{K}_A\,(p)\right) = \tilde{K}_A\,(p) \right\|$.

For every pointed n-species K, the identity morphism on K is given by the term $\lambda A.\left(\lambda k.k, \lambda p.|\ \mathsf{refl}_{\tilde{K}_A(p)}\ |\right)$. Composition of two morphisms $f : K_1 \Rightarrow_* K_2$, $g : K_2 \Rightarrow_* K_3$, is given by mapping each $A : \mathsf{FinSet}$ to a pair $(g_A \circ f_A, \mathcal{P}_A)$. To construct the family of paths $\mathcal{P}_A$, we first observe that since it is propositionally truncated, it is evidently an h-prop, and we can therefore eliminate the truncated paths $\tilde{f}_A\,(q)$, $\tilde{g}_A\,(q)$. Finally, $p_A\,(q)$ is constructed by truncating the composite of the underlying path of $\tilde{f}_A\,(q)$, lifted (by congruence) over the function g_A, with the underlying path of $\tilde{g}_A\,(r)$.

Lemma 4. *Given pointed* n*-species* $K_1 : *_n\text{-}\mathit{Species}_i$, $K_2 : *_n\text{-}\mathit{Species}_j$*, the corresponding hom-type* $K_1 \Rightarrow_* K_2$ *is of homotopy level* $\mathit{max}\,(j, 0)$.

Proof. A full proof is provided in our cubical agda formalisation. The key observations are that every j-type is of homotopy level $\max(j, 0)$ and every h-prop is of homotopy level $\max(j, 0)$. □

Proposition 10. *For any two maps between pointed n-species $f, g : K_1 \Rightarrow_* K_2$, a path*

$$h : \prod_{(A:\mathit{FinSet})} \prod_{(k:K_1(A))} f_A(k) = g_A(k),$$

between their underlying species morphisms, is sufficient to construct a path $p : f = g$.

Proof. By function extensionality, we can construct a path $f = g$ from a family of paths $q : \prod_{(A:\text{FinSet})} f(A) = g(A)$. We recall a known HoTT theorem that for any type $X : \mathcal{U}$, and family of h-props $Y : X \to \mathcal{U}^{\leq -1}$, the path space of $\sum_X Y$ is equivalent to the path space of X. Consequently, it is sufficient to construct a path $r : f_A = g_A$, which follows from application of function extensionality to $h(A)$. □

Proposition 11. *The type $*_1$-Species$_i$ is isomorphic to the type*

$$*_\top\text{-}\mathit{Species}_i := \sum_{(K:\mathit{FinSet}\to\mathcal{U}^{\leq i})} K(\top),$$

where $\top$: FinSet is the finite set trivially constructed from $\top : \mathcal{U}$.

Proof. We observe that our definition of the type of n-pointed species is evidently equivalent to the type $\sum_{(K:\text{FinSet}\to\mathcal{U}^{\leq i})} \left(\prod_{(A:\text{FinSet})} (n = |A|) \to K(A)\right)$. By definition of isomorphism between $\sum$-types, it is sufficient to prove there exists an isomorphism

$$K(\top) \Leftrightarrow \prod_{(A:\text{FinSet})} (1 = |A|) \to K(A).$$

By composition with the isomorphism induced by (dependent) currying and uncurrying we can rewrite the above isomorphism as

$$K(\top) \Leftrightarrow \prod_{\left(\sum_{(A:\text{FinSet})} 1=|A|\right)} K(A).$$

It is known in HoTT that for any type family $Y : X \to \mathcal{U}$, if $X : \mathcal{U}$ is contractible with centre $c : A$, then there is an isomorphism of type $B(c) \Leftrightarrow \prod_{(a:A)} B(a)$. Therefore, the required isomorphism follows from Proposition 8, which asserts that the type $\sum_{(A:\text{FinSet})} 1 = |A|$ is contractible with centre $\top$. □

Proposition 12. *The first projection of the type $*_n$-Species$_0$, which forgets the point, has a left adjoint.*

Proof. The following proof is an internalisation of the proof for Proposition 4, in HoTT. For every $n : \mathbb{N}$ and $K : \text{FinSet} \to \mathcal{U}^{\leq 0}$, we define the free n-pointed species, K_{*n} as the species which maps A : FinSet to the type $K(A) \uplus (n = |A|)$, with the pointing function given by the right injection $\mathbf{inr} : (n = |A|) \to K(A) \uplus (n = |A|)$. The unit of this construction is given by the left injection $\mathbf{inl} : K(A) \to K(A) \uplus (n = |A|)$.

To prove our construction is indeed the left adjoint, it is sufficient to show that for every $K : \text{FinSet} \to \mathcal{U}_0$, $*K$: $*_n$-Species$_0$, and every species morphism $f : \prod_{(A:\text{FinSet})} K(A) \to *K(A)$, there is a unique pointed species map $f^* : K_{*n} \Rightarrow_* *K$, such that for every A : FinSet, $k : K(A)$, there is a path

$$\text{comm}_{A,k} : f_A^*(\mathbf{inl}(k)) = f(A, k).$$

We define f_A^* as the copairing $[f(A), *\tilde{K}_A]$. Preservation of the point is witnessed by the term

$$\tilde{f_A^*} := \lambda p.|\ \text{refl}_{*\tilde{K}_A(p)}\ |.$$

The path $\text{comm}_{A,k}$ is trivially given by reflection on $f(A, k)$. To prove uniqueness of f^*, it is necessary to show that for any $g : K_{*n} \Rightarrow_* *K$, and family of paths $\text{comm}'_{A,k} : g_A(\mathbf{inl}(k)) = f(A, k)$, there is a family of paths $\mathcal{P}_{A,*k} : g_A(*k) = f_A^*(*k)$, for every $*k : K(A) \uplus (n = |A|)$. By induction on $*k$, if we have a term $\mathbf{inl}(t)$ for some $t : K(A)$, then $\mathcal{P}_{A,\mathbf{inl}(t)} := \text{comm}'_{A,t}$. Otherwise, given a term $\mathbf{inr}(p)$ for some $p : n = |A|$, then $\mathcal{P}_{A,\mathbf{inr}(p)}$ is given by elimination of the truncated term $\tilde{g}_A(p)$. We note that elimination is possible because the codomain of g_A and f_A^* are h-sets, and thus the type $g_A(\mathbf{inr}(p)) = f_A^*(\mathbf{inr}(p))$ is an h-prop. □

Definition 7. *A discretely-finite container is given by a shape $S : \mathcal{U}$ and an S-indexed family of finite sets $P : S \to$ FinSet. As usual, we denote such a container by $S \rhd P$.*

Theorem 13. *There exists an isomorphism between the type of discretely-finite containers and the type of combinatorial species. Concretely, the type*

$$(\mathit{FinSet} \to \mathcal{U}) \Leftrightarrow \sum_{(S:\mathcal{U})} S \to \mathit{FinSet},$$

is inhabited.

Proof. The proof of this isomorphism follows from the well-known equivalence between A-indexed type families and fibrations over A of the type $\sum_{(X:\mathcal{U})} X \to A$. We provide a simple sketch of this proof. The forward map sends each $B : A \to \mathcal{U}$ to the type $\sum_{(a:A)} B(a)$ together with the first projection map. The inverse map sends each pair of a type $X : \mathcal{U}$ and function $f : X \to A$ to the type family which maps each $a : A$ to the fibre of f over a, i.e. the type $\sum_{(x:X)} f(x) =_A a$. □

V. Internal Operads

Recall that an operad extends a combinatorial species K : FinSet $\to \mathcal{U}$ with an associative family of composition maps

$$\circ_{n,ns} : K(n) \times K(ns_1) \times K(ns_2) \times \ldots \times K(ns_n) \to K(\Sigma\ ns),$$

which is unital with respect to a distinguished element $e : K(\top)$. However, this classical definition implicitly assumes that every finite set, n : FinSet, comes equipped with a total ordering. In a constructive setting our choice of finiteness dictates whether every finite set has a distinguished total order. In particular, for any given Bishop-finite set, A : FinSet, of cardinality $n : \mathbb{N}$, it is not possible to provide an explicit total

order $I : A \Leftrightarrow \mathrm{Fin}_n$, since the underlying isomorphism is propositionally truncated.

To define the notion of a compositional structure over a combinatorial species in HoTT, we first provide an internal definition of the substitution product of two species.

Definition 8. *The substitution product of two species $K_1, K_2 : \mathit{FinSet} \to \mathcal{U}$ is given by an inductive family $K_1 \circ K_2 : \mathit{FinSet} \to \mathcal{U}$ with a single constructor*

$$\textbf{\textit{group}} : \prod_{(A:\mathit{FinSet})} \prod_{(B:[\![A]\!] \to \mathit{FinSet})} K_1(A) \to \left(\prod_{(a:[\![A]\!])} K_2(B(a)) \right) \to K_1 \circ K_2 \left(\sum_A B \right).$$

While we do not give the proof here, it can be shown that the substitution product is monoidal on combinatorial species of the form $K : \mathrm{FinSet} \to \mathcal{U}^{\leq 0}$. The unit of the substitution product is given by the species which maps each $A : \mathrm{FinSet}$ to $\top : \mathcal{U}^{\leq 0}$ if $|A| = n$, otherwise to $\bot : \mathcal{U}^{\leq 0}$.

Our following internalisation of an operad, in HoTT, can be summarised as the definition of a monoid with respect to the substitution product. In particular, we start by defining the *compositional structure* over a species as follows:

Definition 9. *Given a species $K : \mathit{FinSet} \to \mathcal{U}$, a compositional structure over K is a species morphism $f : K \circ K \Rightarrow K$. Equivalently, a compositional structure over K is given by a family of maps*

$$\mathit{Comp}_{K,A,B} := K\ A \to \left(\prod_{(a:[\![A]\!])} K\ (B\ a) \right) \to K\ \left(\sum_A B \right),$$

for every $A : \mathit{FinSet}$, $B : [\![A]\!] \to \mathit{FinSet}$.

Given a pointed species $(K, e) : *_\top\text{-Species}$, an operad over (K, e) is an associative compositional structure which is unital with respect to e. To complete our internalisation of the definition of an operad, we require path terms which witness the associativity and unitality of a given compositional structure.

We start by observing that by application of Lemma 1, and by closure of finite sets under $\sum$ (Proposition 7), the evident paths witnessing associativity and unitality of $\sum$, for unit $\top : \mathcal{U}$, can be lifted to paths between the corresponding finite sets. We denote these paths by $\Sigma\mathrm{Assoc}_{A,B,C}$, $\Sigma\mathrm{Idl}_A$ and $\Sigma\mathrm{Idr}_A$ respectively, for $A : \mathrm{FinSet}$, $B : [\![A]\!] \to \mathrm{FinSet}$, $C : \prod_{(a:[\![A]\!])} [\![B(a)]\!] \to \mathrm{FinSet}$.

Associativity of a compositional structure $(- \bullet_{X,Y} -) : \mathrm{Comp}_{K,X,Y}$ is witnessed by a family of paths

$$\begin{aligned} \mathrm{isAssoc}_{A,B,C}(- \bullet -, a, b, c) := \\ (a \bullet_{A,B} b)\ \bullet_{(\sum_A B), C}\ c =^{\Sigma\mathrm{Assoc}_{A,B,C}} \\ a \bullet (\lambda a.b(a)\ \bullet_{B(a),\ C(a)}\ c(a)), \end{aligned}$$

for all terms $a : K(A)$, $b : \prod_{(a:[\![A]\!])} K(B(a))$, $c : \prod_{(a:[\![A]\!])} \prod_{(b:[\![B(a)]\!])} K(C(a,b))$. In a similar fashion, left and right unitality are witnessed by paths

$$\mathrm{Idl}_{e,A}(- \bullet -, a) := e \bullet_{\top, \lambda i.A} (\lambda i.a) =^{\Sigma\mathrm{Idl}_A} a,$$
$$\mathrm{Idr}_{e,A}(- \bullet -, a) := a \bullet_{A, \lambda a.\top} (\lambda i.e) =^{\Sigma\mathrm{Idr}_A} a.$$

The types witnessing associativity and unitality, of a compositional structure over a species, give rise to the following internalisation of an operad in HoTT:

Definition 10. *A (non-symmetric) operad is a term of type*

$$\begin{aligned} \mathit{Operad} := \sum_{((K,e):*_\top\text{-}\mathit{Species}_1)} \sum_{((-\bullet-):\mathit{Comp}_K)} \\ \mathit{isAssoc}(- \bullet -) \times \mathit{Idl}_e(- \bullet -) \times \mathit{Idr}_e(- \bullet -). \end{aligned}$$

Given an operad $O : \mathit{Operad}$, we will often write $O(A)$ to denote the application of its underlying species map to a finite set A.

Definition 11. *Recall the definition of a pointed species map given in Definition 6. An operad map between two operads $(K_1,\ e_1,\ -\bullet^1-,\ p_1)$, $(K_2,\ e_2,\ -\bullet^2-,\ p_2) : \mathit{Operad}$ is given by a morphism $f : K_1 \Rightarrow_* K_2$ between the underlying pointed species which preserves operad composition. Preservation of operad composition is witnessed by a family of paths*

$$\begin{aligned} f^{\mathit{comp}}_{A,B}(a,b) : f_{\sum_A B}\left(a \bullet^1_{A,B} bs\right) = \\ f_A(a) \bullet^2_{A,B} \left(\lambda x.f_{B(x)}(b(x))\right), \end{aligned}$$

for all $A : \mathit{FinSet}$, $B : [\![A]\!] \to \mathit{FinSet}$, $a : K(A)$ and $b : \prod_{(a:[\![A]\!])} K(B(a))$.

The identity operad map is given by the identity morphism on species, for which preservation of identity and composition is trivially given by reflection. Given operad maps $f : O_1 \Rightarrow O_2$ and $g : O_2 \Rightarrow O_3$ their composite, $g \circ f : O_1 \Rightarrow O_3$, is given by composition of the underlying morphisms over pointed species. The path witnessing the composite preserves operad composition is constructed by lifting the paths $f^{\mathrm{comp}}_{A,B}(a,b)$ over g and composing with $g^{\mathrm{comp}}_{A,B}\left(f_A(a), \lambda a.f_{B(a)}(b(a))\right)$.

Lemma 5. *The hom-type $O_1 \Rightarrow O_2$ between two operads $O_1, O_2 : \mathit{Operad}$ is an h-set.*

Proof. We start by observing that the hom-type between the underlying pointed species of O_1 and O_2 is an h-set (Proposition 4). Since the type witnessing preservation of operad identity characterises a family of paths in $O_2(\sum_A B)$, its type can readily be shown to be an h-prop, and thus an h-set. □

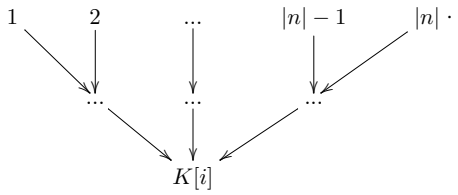
Proposition 14. *The path space $f = g$, between any two operad maps $f, g : O_1 \Rightarrow O_2$, is isomorphic to the type $\prod_{(A:\mathit{FinSet})} O_1(A) = O_2(A)$.*

Proof. As described in the proof of Lemma 5, the type witnessing preservation of operad identity is an h-prop. Therefore, the path space $f = g$ is equivalent to the path space between the underlying pointed species morphisms of f and g. By Proposition 10 we can construct a path of type $f = g$ from

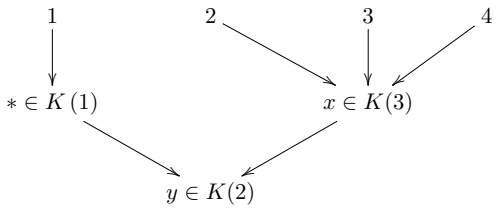
a term of type $\prod_{(A:\text{FinSet})} O_1(A) = O_2(A)$. The inverse construction takes advantage of the underlying structure of paths in Σ-types, i.e. a dependent pair of paths, to construct an element of $\prod_{(A:\text{FinSet})} O_1(A) = O_2(A)$ from a path $f = g$. Since $O_1 \Rightarrow O_2$ is an h-set (Lemma 5), it follows that this two-way construction is an isomorphism. □

VI. The Free Operad

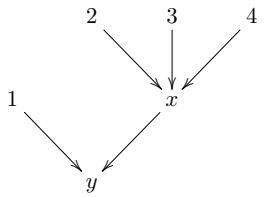
The free operad on a combinatorial species $K : \mathbb{B} \to \text{Set}$, where Set is the category of (small) sets and functions, is given by the left adjoint to the forgetful functor from the category of operads to the category of species. For each $n \in \mathbb{B}$, define $\mathbb{T}_{K,n}$ to be the set of labelled, rooted trees whose leaves are in bijection with n, and where each vertex V is labelled with an element of $K[i]$, where i is the number of input edges to V:



Given the free pointed species K_{1+}, the free operad is a quotient of the set $\mathbb{T}_{K_{1+},n}$, in which each vertex labelled by the freely adjoined point is collapsed. For example, the tree



reduces to



Our internalisation of the 'free operad over a species' is similarly defined in two steps. The first is the definition of an inductive family $\mathbb{T}_K : \text{FinSet} \to \mathcal{U}$ of finitely branching, K-labelled, rooted trees. We then introduce path constructors to define a quotient of $\mathbb{T}_K$ for which tree composition is both associative and unital.

Definition 12. *Given a species $K : \mathit{FinSet} \to \mathcal{U}$, we define the family $\mathbb{T}_K : \mathit{FinSet} \to \mathcal{U}$ of finitely branching, K-labelled, rooted trees as the inductive family with data constructors*

$$\begin{aligned}
&\textbf{\textit{unit}} : \mathbb{T}_K(\top),\\
&\textbf{\textit{corolla}} : \prod_{(A:\mathit{FinSet})} K(A) \to \mathbb{T}_K(A),\\
&\textbf{\textit{graft}} : \prod_{(A:\mathit{FinSet})} \prod_{(B:[\![A]\!]\to\mathit{FinSet})} \\
&\qquad \mathbb{T}_K(A) \to \left(\prod_{(a:[\![A]\!])} \mathbb{T}_K(B\ a)\right) \to \mathbb{T}_K\left(\sum_A B\right).
\end{aligned}$$

The **unit** constructor corresponds to a rooted tree with a single input edge and whose label is the freely adjoined point. As evidenced by its name, the **corolla** constructor builds a tree with input edges A : FinSet, a single output edge, and label $k : K(A)$. Given a tree with input edges A : FinSet and, for every $a : [\![A]\!]$, a tree with input edges $B(a)$: FinSet, the **graft** constructor 'composes' trees together by gluing the outputs of the latter trees to the inputs of the former.

Definition 13. *The family $\mathit{Free}_K : \mathit{FinSet} \to \mathcal{U}$ is a quotient of $\mathbb{T}_K$ for which the **graft** constructor is associative and unital with respect to **unit**. Explicitly, we define Free_K by extending the inductive definition of $\mathbb{T}_K$ with path constructors*

$$\begin{aligned}
&\textbf{\textit{idl}} : \prod_{(A:\mathit{FinSet})} \prod_{(t:\mathbb{T}_K(A))} \textbf{\textit{graft}}(\top, \lambda x.A, \mathit{unit}, \lambda x.t) =^{\Sigma \mathit{Idl}_A} t\\
&\textbf{\textit{idr}} : \prod_{(A:\mathit{FinSet})} \prod_{(t:\mathbb{T}_K(A))} \textbf{\textit{graft}}(A, \lambda a.\top, t, \lambda a.\mathit{unit}) =^{\Sigma \mathit{Idr}_A} t,
\end{aligned}$$

together with a family of constructors

$$\begin{aligned}
&\textbf{\textit{assoc}}_{A,B,C,t,ts,tss} :\\
&\textbf{\textit{graft}}\left(\sum_A B,\ C,\ \textbf{\textit{graft}}(A,B,t,ts),\ tss\right) =^{\Sigma \mathit{Assoc}_{A,B,C}}\\
&\textbf{\textit{graft}}\left(A,\ \sum_B C,\ t,\ \textbf{\textit{graft}}(B,C,ts,tss)\right).
\end{aligned}$$

Since we are only considering coherences for operads whose operations are h-sets, it is necessary to impose that the operations of the free operad are h-sets, by including a set truncation constructor

$$\textbf{\textit{trunc}} : \prod_{(A:\mathit{FinSet})} \mathit{isSet}(\mathbb{T}_K(A)).$$

Proposition 15. *For any species $K : \mathit{FinSet} \to \mathcal{U}$, the family $\mathit{Free}_K : \mathit{FinSet} \to \mathcal{U}$ is isomorphic to the free operad on K.*

Proof. The higher inductive family Free_K is trivially isomorphic to the operad defined by its constructors. Our proof follows by showing that this construction satisfies the property of being left adjoint to the forgetful functor which maps each operad to its underlying species.

The higher inductive family Free_K comes equipped with a (uniquely-defined) eliminator which, given an operad O : Operad, lifts a species morphism $f : K \Rightarrow O$ to an operad morphism $\langle - \rangle_f : \text{Free}_K \Rightarrow O$. Intuitively, $\langle - \rangle_f$ maps a tree to an operation of O by mapping each labelled vertex of the tree to the operation specified by f and then applying operadic

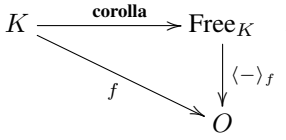
composition in O as specified by the structure of the tree. Concretely, we define the underlying species morphism of $\langle - \rangle_f$, by induction over the constructors of Free_K, and then prove that our definition is an operad map.

The species morphism $\langle - \rangle_f$ maps the **unit** tree to the point (unit) of O, and **corolla** (A, k) to $f\,(k) : O\,(A)$. A composite tree **graft** (A, B, t, ts) is mapped to the term

$$\langle\ t\ \rangle_{f,A} \bullet_{O,A,B} (ts \circ \langle - \rangle_{f,B}) : O\left(\sum_A B\right).$$

Elimination of the path constructors of Free_K follows from the paths witnessing that $(- \bullet_O -)$ respects operadic laws, and that operations of O are h-sets. The proof that $\langle - \rangle_f$ extends to an operad morphism follows definitionally from its action on trees built by **unit** and **graft**.

By taking the **corolla** constructor as the unit of our free construction, we observe that the following diagram commutes



Furthermore, it can be shown that $\langle - \rangle_f$ is the unique map making this diagram commute. A proof of the uniqueness of $\langle - \rangle_f$ is given in our cubical Agda formalisation. □

VII. Algebras

As is typical of categorical structures, to realise the abstract collection of operations defined by an operad as concrete operations on a given type, we introduce an internalisation of the notion of an operad algebra. In particular, a concrete operation on an h-set $X : \mathcal{U}$, is to be understood as a function of type $(\llbracket A \rrbracket \to X) \to X$, for $A : \text{FinSet}$. For any given type $X : \mathcal{U}$, functions of the form $(\llbracket A \rrbracket \to X) \to X$ give rise to an evident species, which in turn has an important operadic structure. The operad induced over this collection of functions is termed the *endomorphism* operad over X. As with our previous internalisations, the following definitions differ from the typical classical definition of the endomorphism operad in the absence of a total order on finite sets.

Definition 14. *The endomorphism operad over an h-set $X : \mathcal{U}^{\leq 0}$ is given by the 1-pointed species $(\llbracket - \rrbracket \to X) \to X$ with point $\lambda f.f\,(*)$, together with a composition function which maps every A : FinSet, $B : \llbracket A \rrbracket \to$ FinSet and terms*

$$f : (\llbracket A \rrbracket \to X) \to X,\ g : \prod_{(a:\llbracket A \rrbracket)} (\llbracket B\,(a) \rrbracket \to X) \to X,$$

to the composite

$$\lambda h.f\,(\lambda a.g\,(a, (\lambda b.h\,(a, b)))) : \left(\left[\!\left[\sum_A B\right]\!\right] \to X\right) \to X.$$

Proposition 16. *The composition function of the endomorphism operad (Definition 14) satisfies the required unitality and associativity conditions, as specified in Definition 10.*

Proof. Given types $X, Y, Z : \mathcal{U}$, path $p : X = Y$, and function $f : (Y \to Z) \to Z$, we can construct a path

$$\mathcal{E}_p\,(f) : \lambda h.f\,(h \circ p^{\leftarrow}) =^p f.$$

By path induction (i.e. J-elimination), it is sufficient to construct a path of type

$$\textbf{refl}^*_X\,(\lambda h.g\,(h \circ \textbf{refl}^{\leftarrow})) = g,$$

for every $g : (X \to Z) \to Z$. By application of function extensionality, and the computation rule of the J-eliminator, we can construct the desired path $\mathcal{E}_p\,(f)$. The proofs of left and right unitality, and associativity follow from the terms $\mathcal{E}_{\Sigma\text{Idl}}$, $\mathcal{E}_{\Sigma\text{Idr}}$, and $\mathcal{E}_{\Sigma\text{Assoc}}$, respectively. □

Definition 15. *Following Definition 10 and Proposition 14, an algebra over an operad O : Operad is an h-set $X : \mathcal{U}^{\leq 0}$, together with an operad map from O to Endo_X. Concretely, we internalise algebras over O as terms of the type*

$$\mathit{Algebra}_O := \sum_{(X:\mathcal{U}^{\leq 0})} (O \Rightarrow \mathit{Endo}_X).$$

Proposition 17. *Given K : FinSet $\to \mathcal{U}^{\leq 0}$, there is an isomorphism between the type of algebras over the free operad Free_K and the type*

$$\sum_{(X:\mathcal{U}^{\leq 0})} \prod_{(A:\mathit{FinSet})} K\,(A) \to (\llbracket A \rrbracket \to X) \to X.$$

*Namely, to define an algebra over Free_K it is sufficient to define its action on the **corolla** constructor.*

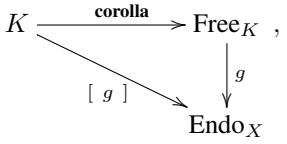
Proof. Given $X : \mathcal{U}^{\leq 0}$, together with a species morphism

$$f : \prod_{(A:\text{FinSet})} K\,(A) \to (\llbracket A \rrbracket \to X) \to X,$$

the (unique) eliminator of the higher inductive family Free_K (see proof of Proposition 15) constructs an operad map $\langle - \rangle_f : \text{Free}_K \Rightarrow \text{Endo}_X$. For any operad map $g : \text{Free}_K \Rightarrow \text{Endo}_X$, we define the inverse map by

$$[\ g\ ]\,(A, k, h) = g\,(A, \textbf{corolla}\,(A, k), h)$$

for all A : FinSet, $k : K\,(A)$, $h : \llbracket A \rrbracket \to X$. By definition, $\langle - \rangle_f$ maps the term **corolla** (A, k) to $f\,(A, k)$. It follows that a path of type $[\ \langle - \rangle_f\ ] = f$ is simply given by reflection on f. In the other direction, for any operad map $g : \text{Free}_K \Rightarrow \text{Endo}_X$ the diagram



evidently commutes (by definition of $[-]$). By the uniqueness property of the eliminator $\langle - \rangle_-$ it follows that $\langle - \rangle_{[g]} = g$. □

By application of the isomorphism established by Theorem 13, given an h-set $S : \mathcal{U}^{\leq 0}$, every discretely-finite container $S \triangleright P$ gives rise to a free construction of an operad. Naturally, this is simply the free operad over the species defined by mapping every A : FinSet to the fibre of P over A. We denote this operad by $\text{Free}_{(S \triangleright P)}$. Intuitively, the underlying species

of this operad maps every $A : \text{FinSet}$ to the set of (composite) constructors with positions indexed by A.

Theorem 18. *Given a discretely finite container $S \triangleright P$, the type of operad algebras over $\text{Free}_{(S \triangleright P)}$ is isomorphic to the type*

$$\sum_{(X:\mathcal{U}^{\leq 0})} \prod_{(s:S)} (\llbracket P\,(s) \rrbracket \to X) \to X.$$

That is, the type of algebras over the free operad on a discretely finite container is isomorphic to the type of algebras, restricted to h-sets, over that container.

Proof. By Proposition 17, given an h-set $X : \mathcal{U}^{\leq 0}$, the type of operad maps $\text{Free}_{(S \triangleright P)} \Rightarrow \text{Endo}_X$ is isomorphic to the type

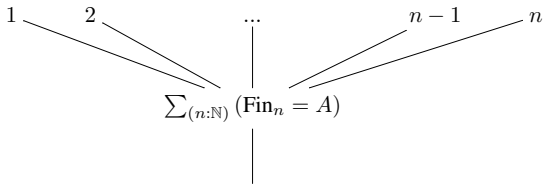
$$\prod_{(A:\text{FinSet})} \left(\sum_{(s:S)} P\,(s) = A \right) \to (\llbracket A \rrbracket \to X) \to X.$$

By dependent currying, simple rearrangement of inputs, and application of Lemma 6 this type can be shown to be isomorphic to

$$\prod_{(s:S)} \prod_{(A:\text{FinSet})} (P\,(s) = A) \to (\llbracket P\,(s) \rrbracket \to X) \to X.$$

It can be shown that for any $A' : \text{FinSet}$, a finite set $A : \text{FinSet}$ together with a path $A' = A$ form a contractible pair. Consequently, for any $s : S$ the type $\sum_{(A:\text{FinSet})} P\,(s) = A$ is contractible. Applying this isomorphism, we can establish an isomorphism between the type above and $\prod_{(s:S)} (\llbracket P\,(s) \rrbracket \to X) \to X$. □

For example, consider the finitary container, $\text{List} := \mathbb{N} \triangleright \text{Fin}$, whose extension gives rise to the finite lists. The free operad over this container is isomorphic to the type of trees which can be built by grafting together corollas of the form



for any $A : \text{FinSet}$. An operad algebra over $\text{Free}_{(\mathbb{N} \triangleright \text{Fin})}$ is given by an h-set $X : \mathcal{U}^{\leq 0}$, together with an operad map $f : \text{Free}_{(\mathbb{N} \triangleright \text{Fin})} \Rightarrow \text{Endo}_X$. By Theorem 18, f is equivalently given by a function of the type

$$\prod_{(n:\mathbb{N})} (\text{Fin}_n \to X) \to X.$$

In this way, the type of algebras over $\text{Free}_{(\mathbb{N} \triangleright \text{Fin})}$ are equivalently the type of algebras over the Lawvere theory encoded by the ambient type theory.

Lemma 6. *For any type $X : \mathcal{U}$, family $Y : X \to \mathcal{U}$, and term $y : Y$, there is an isomorphism*

$$\prod_{(y':Y)} y = y' \to Z\,(y) \Leftrightarrow \prod_{(y':Y)} y = y' \to Z\,(y')\,.$$

Proof. We construct the forward and backwards map by substitution along a path $p : y = y'$ and its inverse $p^- : y' = y$, respectively. The proof terms witnessing that this is indeed an isomorphism follow from the known result that substitution along a path followed by substitution along its inverse is the identity map. □

VIII. Examples

To illustrate how our internalisation of operads can be used to describe suitable compositional structures in HoTT, we introduce two such structures which can be captured by our notion of HoTT operads. The first structure, namely the coendomorphism operad, is of particular importance to the further study of operads and their related structures. The second outlined example, the 'substitution operad', illustrates how operads can be used for the purpose of generic programming, in a similar fashion to the theory of containers.

Coendomorphism Operad. For every h-set $X : \mathcal{U}^{\leq 0}$ consider the species

$$\text{CoEndo}_X\,(A) := X \to \llbracket A \rrbracket \to X,$$

which inverts the direction of the underlying species of the endomorphism operad. Similar to Endo_X, the species CoEndo_X forms an operadic structure. The unit of this operad is given by the constant function, i.e. $\lambda x.\lambda a.x$. Given $A : \text{FinSet}$ and $B : \llbracket A \rrbracket \to \text{FinSet}$, the composition map of Endo_X^{-1} composes two terms $f : X \to \llbracket A \rrbracket \to X$ and $g : \prod_{(a:\llbracket A \rrbracket)} X \to \llbracket B\ a \rrbracket \to X$, by constructing the term

$$\lambda x.\lambda c.f\,(g\,(\pi_1\,(c)\,, x, \pi_2\,(c))) : X \to \left(\sum_{(a:\llbracket A \rrbracket)} \llbracket B\ a \rrbracket \right) \to X.$$

The proof terms witnessing associativity and unitality follow in a similar manner to that of the endomorphism operad.

As might be expected, the coendomorphism operad serves a dual role, in the theory of operads, to that of the endomorphism operad. Namely, it can be used to encode the type of *coalgebras* over an operad $O : \text{Operad}$ as

$$\text{Coalgebra}_O := \sum_{(X:\mathcal{U}^{\leq 0})} O \Rightarrow \text{CoEndo}_X.$$

Substitution Operad. Given a type $X : \mathcal{U}$, every monoidal operation, $(- \cdot -) : X \to X \to X$, with unit $e : X$, gives rise to a particular 'substitution' operad. Intuitively, we can understand the operation $(- \cdot -)$ as describing the substitution 'strategy'. The underlying species of the substitution operad over $(- \cdot -)$ maps each finite set $A : \text{FinSet}$ to the type of functions $\llbracket A \rrbracket \to X$. Given $B : \llbracket A \rrbracket \to \text{FinSet}$, the composition map composes functions $f : \llbracket A \rrbracket \to X$, $g : \prod_{(a:\llbracket A \rrbracket)} \llbracket B\ a \rrbracket \to X$ by constructing the term

$$\lambda c.f\,(\pi_1\,(c)) \cdot g\,(\pi_1\,(c)\,, \pi_2\,(c)) : \left[\!\!\left[\sum_A B \right]\!\!\right] \to X.$$

The unit of the substitution operad is simply given by the term $\lambda x.e : \llbracket \top \rrbracket \to X$.

A concrete example for understanding the substitution operad can be given by instantiating X as Bool, the 2-element type of booleans, and defining $(- \cdot -)$ as the *XNOR* operation. The unit of the *XNOR* operation is **True** or 1. To illustrate the composition map of this substitution operad, we can consider each operation $[\![A]\!] \to$ Bool as corresponding to a finite list of $|A|$ booleans. Strictly speaking, functions of the type $[\![A]\!] \to$ Bool do not impose an ordering on the A-indexed collection of booleans, and therefore do not correspond to lists. However, the following illustration can be made precise by simply replacing 'list' with 'finitely-indexed collection'.

We compose a list l of booleans with a list ls of lists of booleans by first mapping every list in ls to itself if the boolean at the corresponding position in l is 1, otherwise we invert every boolean value in that list. Finally, we flatten the resulting list of lists. We can interpret the operations of the *XNOR* substitution operad as logic gates which output true for a desired input combination and false otherwise, with composition given by the obvious plugging of inputs into outputs.

IX. Related Work

There have been several generic representations of datatypes developed in the type theory literature. The idea of separating structure from data first arose in the work on so-called shapely types [11]. More recently, the theory of containers was developed to capture the idea of the strictly positive types [1]. The more general presentation of 'indexed' containers was later developed to capture inductive families [12]. Containers have been shown to be closed under finite products and exponentials [13], and have a sensible notion of derivative [14]. The theory of containers also captures the shapely types. In particular, shapely type constructors are precisely given by the extensions of discretely-finite containers.

A key aspect of our theory of internal operads is the notion of a finite set, and the corresponding choice of a suitable definition of finiteness. Such a choice only manifests in constructive mathematics, where various classically equivalent notions of finiteness are notably distinct. In this paper we adopt the notion of finiteness used in previous work on internalising the theory of combinatorial species in HoTT [3], namely Bishop-finiteness. However, there has also been work on internalising 'enumerated' types and Kuratowski finite sets in the framework of HoTT [15, 8]. Adopting a different notion of finiteness impacts both which collection of operations are definable as a species, and which proofs about operads can be internalised.

The theory of combinatorial species were first introduced to unify existing approaches for analysing generating functions of discrete structures [16]. Intuitively, the idea of a species arose from the idea of building a structure from a finite collection of labels. Our internalisation of operads in HoTT can be seen as an extension of previous work on internalising combinatorial species [3]. In particular, Yorgey internalises combinatorial species to capture specific classes of data types, analogous to shapely types and the theory of containers.

The theory of operads first originated in the field of algebraic topology [17, 18] for describing operations on iterated loop-spaces.

X. Further Work

In this paper we have demonstrated how operads can be internalised in homotopy type-theory, and how this gives rise to a generic calculus of operations. Notably, we show how discretely finite containers, which represent inductive data types whose constructors have a finite arity, give rise to a natural operadic interpretation. Intuitively this provides an operation-centric approach to inductive types, in contrast to the traditional data-centric approach. Our results open up a new line of research on investigating the properties and applications of operads from a type theoretic perspective.

A natural extension of our work is a generalisation from species and operads to their 'coloured' counterparts. A 'coloured' species extends the standard definition of a combinatorial species by additionally indexing over the 'colours' of the inputs and output of operations. In an identical fashion to that of an operad, a 'coloured' operad extends a coloured species with a compositional structure together with unitality and associativity coherence conditions. Intuitively, the 'coloured' variants of species and operads give rise to a *typed* calculus of operations. We expect that the relationship between finitary containers and combinatorial species generalises to their indexed and coloured variants.

Traditionally, a combinatorial species is defined as a presheaf over the 'groupoid of finite sets and bijections'. In constructive mathematics, it is typical to interpret finite as 'Bishop-finite', which most closely resembles the classical definition. However, many classically equivalent notions of finiteness are constructively distinct. An example we give in this paper, in our introduction to internalising finite sets, is that of Kuratowksi or 'finitely-indexed' sets. In particular, we might consider how a choice of finiteness influences which algebraic structures are expressible as a family over finite sets.

As is typical for algebraic structures, operads have a natural dual in 'cooperads'. Cooperads are defined by the evident dualisation of operads, i.e. by reversing the direction of the composition and unit maps. Intuitively, a cooperad over a collection of operations describes a means by which an operation can be partitioned into 'smaller' operations, in a manner that is counital and coassociative. In this way, cooperads are to operads as coinductive types are to inductive types. An internalisation of the theory of cooperads, in HoTT, would seem to follow by a trivial dualisation of the definitions given in this paper. However, an important asymmetry arises when considering an appropriate implementation of the cofree cooperad. From the definition of the free operad, we can infer that the cofree cooperad should be isomorphic to a type $\mathbb{G}$ of graphs, in which each node has a label, a distinguished output edge, and a finite set of input edges. Equivalently, $\mathbb{G}$ is the type of coinductive (infinite), finitely-branching, labelled trees. We postulate that, in contrast to its inductive counterpart, counitality and coassociativity can be proven for the 'de-composition' operation of $\mathbb{G}$-graphs without the need for quotients.

REFERENCES

[1] M. Abbott, "Categories of containers," Ph.D. dissertation, University of Leicester, 2003.

[2] M. Abbott, T. Altenkirch, and N. Ghani, "Containers: Constructing strictly positive types," *Theoretical Computer Science*, vol. 342, no. 1, pp. 3 – 27, 2005, applied Semantics: Selected Topsics. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0304397505003373

[3] B. A. Yorgey, "Combinatorial species and labelled structures," Ph.D. dissertation, University of Pennsylvania, 2014.

[4] "HoTT Operads Cubical Agda Library," https://tinyurl.com/y5fbrzdb, 2020.

[5] C. Cohen, T. Coquand, S. Huber, and A. Mörtberg, "Cubical type theory: a constructive interpretation of the univalence axiom," *arXiv preprint arXiv:1611.02108*, 2016.

[6] A. Vezzosi, A. Mörtberg, and A. Abel, "Cubical agda: a dependently typed programming language with univalence and higher inductive types," *Proceedings of the ACM on Programming Languages*, vol. 3, no. ICFP, pp. 1–29, 2019.

[7] T. U. F. Program, "Homotopy type theory: Univalent foundations of mathematics," Institute for Advanced Study, Tech. Rep., 2013.

[8] D. Frumin, H. Geuvers, L. Gondelman, and N. v. d. Weide, "Finite sets in homotopy type theory," in *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs*, 2018, pp. 201–214.

[9] D. S. Bridges, "Constructive mathematics: a foundation for computable analysis," *Theoretical computer science*, vol. 219, no. 1-2, pp. 95–109, 1999.

[10] G. M. Kelly, "Many-variable functorial calculus. i." in *Coherence in Categories*, G. M. Kelly, M. Laplaza, G. Lewis, and S. Mac Lane, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1972, pp. 66–105.

[11] C. B. Jay and J. R. B. Cockett, "Shapely types and shape polymorphism," in *European Symposium on Programming*. Springer, 1994, pp. 302–316.

[12] T. Altenkirch, N. Ghani, P. Hancock, C. McBride, and P. Morris, "Indexed containers," *Journal of Functional Programming*, vol. 25, 2015.

[13] T. Altenkirch, P. Levy, and S. Staton, "Higher-order containers," in *Conference on Computability in Europe*. Springer, 2010, pp. 11–20.

[14] M. Abbott, T. Altenkirch, N. Ghani, and C. McBride, "Derivatives of containers," in *International Conference on Typed Lambda Calculi and Applications*. Springer, 2003, pp. 16–30.

[15] T. Coquand and A. Spiwack, "Constructively finite?" in *Contribuciones científicas en honor de Mirian Andrés Gómez*. Universidad de La Rioja, 2010, pp. 217–230.

[16] A. Joyal, "Une théorie combinatoire des séries formelles," *Advances in mathematics*, vol. 42, no. 1, pp. 1–82, 1981.

[17] J. P. May, *The geometry of iterated loop spaces*. Springer, 2006, vol. 271.

[18] J. M. Boardman and R. M. Vogt, *Homotopy invariant algebraic structures on topological spaces*. Springer, 2006, vol. 347.