Summary of the things that I learned

Arnoud van der Leer

CHAPTER 1

Lessons about coq and unimath

When writing coq code, make sure you understand why a proof should work, instead of blindly unfolding and applying Lemmas. That improves the overall quality of the proofs.

A proof closed with `Qed` is opaque, whereas a proof that closes with `Defined` is transparent (i.e. is remembered can be unfolded). Which one is the right one requires some thought.

Only use `destruct` in opaque proofs.

Path induction (or induction on proofs of equality) helps a lot when proving something about a `transportf`.

1. One time checklist

- Check the vscode setting for removing trailing whitespace on save;
- Check the vscode setting for making sure that there is exactly 1 trailing newline on save?

2. Checklist before making a PR

- Make sure that all variables in intros and induction (that are used later) are introduced by name;
- Make sure that `use` is replaced by `apply` and `apply` by `exact` wherever possible;

3. Default API for any `object`

- `object_data`: A definition for the data of the object;
- `make_object_data`: A function to make the object data out of its parts;
- (Coercions from `object_data` to some of its parts;)
- (Explicit functions to access the constituents;)
- `is_object`: A definition for the properties of the object;
- `make_is_object`: A function to make the properties part of the object;
- `make_object`: A function to make the object out of its data and property components;
- `object`:
- (A coercion from the object to its data;)
- (`object_has_property`: Explicit functions to access the properties;)
- `is_object_isaprop`: A lemma that the properties form a proposition;
- `object_eq`: A lemma about sufficient (and necessary) conditions for two terms of type `object` to be equal (usually some conditions on the constituents of `object_data`);

4. Cleaning up code

It is okay to simplify proofs as much as possible. When one wants to step through a proof, they can add `simpl`s to their own liking.

`cbn` is stronger than `simpl`, so it can be good to try and replace `cbn`s with `simpl`s when cleaning up code.

5. Things I was not able to find on my own

```
isweq_iso
weqdnicoprod
transportf_set
```

6. Things I would have liked to know at the start

The fact that if a rewrite does not succeed because a lemma has a slightly different (folded or unfolded) description, we can just force its form like `rewrite (lemma : form1 = form2)`.

Setting just one implicit argument like `action (n := 5)`.

The fact that instead of finishing a (sub)proof with `[tactic]. apply idpath.`, one can finish it with `now [tactic].`.

The pushout of `exact`, `apply`, `refine` and `use`.

7. Things that I would like to have a standard for

- Naming (of folders, files, terms): what abbreviations are allowed? In what order do words appear? How are words separated? How are names capitalized?
- Documentation headers in files.
- In which cases do we want to use unicode?
- In which cases do we want definitions to be (fully) typed?
- In which cases do we want accessor functions for all data and properties?
- What should be documented using comments, and how should they be formatted?

No lines should have trailing whitespace and all files should have a trailing newline.

Week 08

8. Univalent Categories

A univalent category is a category in which the univalence axiom holds. I.e., a category $\mathcal{C}$ in which, for all $A, B \in \mathcal{C}_0$, the canonical map $(A =_{\mathcal{C}} B) \to (A \cong B)$ is an equivalence.

9. Categories

An **n-category** is a category with 0-cells (objects), 1-cells (morphisms), 2-cells (morphisms between morphisms), up to n-cells and various compositions: $A \to B \to C$. $A \xrightarrow{f,g,h} B$, $f \Rightarrow g \Rightarrow h$. $A \xrightarrow{f,g} B \xrightarrow{f',g'} C$, $\alpha : f \Rightarrow g$, and identities $\alpha' : f' \Rightarrow g'$ gives $\alpha' * \alpha : f' \circ f \Rightarrow g' \circ g$. These all need to work together 'nicely'. An ω-category is the same, but all the way up.

A topological space gives a (weak) ω-category. 0-cells are points, 1-cells are paths, 2-cells are homotopies etc. Composition is by glueing. It is a 'groupoid', in the sense that all homotopies of dimension ≥ 1 are invertible. However, glueing is not associative, so it is a 'weak' ω-category.

A category with only one object $\star$ is equivalent to a monoid (with elements being the set $\mathcal{C}(\star, \star)$). A 2-category with only one 0-cell is the same thing as a monoidal category (objects: the 1-cells. Morphisms: the 2-cells). A monoidal category with just one object gives 2 monoid structures on its set of morphisms. These are the same, and commutative.

A **monoid** is a set with a multiplication and a unit. A **monad** on a category $\mathcal{C}$ is a functor $T : \mathcal{C} \to \mathcal{C}$, together with natural transformations $\mu : T \circ T \to T$ (satisfying associativity) and $\eta : 1_{\mathcal{A}} \to T$ (acting as a two-sided unit).

A **presheaf** on a category $\mathcal{A}$ is a functor $\mathcal{A}^{opp} \to \mathbf{Set}$.

Given a category $\mathcal{E}$ and an object $E \in \mathcal{E}_0$, the **slice category** $\mathcal{E}/E$ with objects being the maps $D \xrightarrow{p} E$ and morphisms being commutative triangles.

A **multicategory**, not necessarily the same as an n-category, is a category in which arrows go from multiple objects to one, instead of from one object to one. I.e. it is a category with a class C_0 of objects, for all n, and all $a, a_1, \ldots, a_n \in C_0$, a class $C(a_1, \ldots, a_n; a)$ of 'morphisms', and a composition

$$C(a_1, \ldots, a_n; a) \times C(a_{1,1}, \ldots, a_{1,k_1}; a_1) \times \cdots \times C(a_{n,1}, \ldots, a_{n,k_n}; a_n) \to C(a_{1,1}, \ldots, a_{n,k_n}; a),$$

written $(\theta, \theta_1, \ldots, \theta_n) \mapsto \theta(\theta_1, \ldots, \theta_n)$ and for each $a \in C_0$ an identity $1_a \in C(a; a)$. It must satisfy associativity

$$\theta \circ (\theta_1 \circ (\theta_{1,1}, \ldots, \theta_{1,k_1}), \ldots, \theta_n \circ (\theta_{n,1}, \ldots, \theta_{n,k_n})) = (\theta \circ (\theta_1, \ldots, \theta_n)) \circ (\theta_{1,1}, \ldots, \theta_{n,k_n})$$

and identity

$$\theta \circ (1_{a_1}, \ldots, 1_{a_n}) = \theta = 1_a \circ \theta.$$

A **map of multicategories** is a function $f_0 : C_0 \to C_0'$ and maps $C(a_1, \ldots, a_n; a) \to C(f_0(a_1), \ldots, f_0(a_n); f_0(a))$, preserving composition and identities.

For C a multicategory, a **C-algebra** is a map from C into the multicategory **Set** (with objects $\mathbf{Set}_0$ and maps $\mathbf{Set}(a_1, \ldots, a_n; a) = \mathbf{Set}(a_1 \times \cdots \times a_n; a)$). I.e., for each $a \in C_0$, a set $X(a)$, and for each map $\theta : a_1, \ldots, a_n \to a$, a function $X(\theta) : X(a_1) \times X(a_n) \to X(a)$. An example is, for a multicategory C, to take $X(a) = C(; a)$ (maps from the empty sequence into a).

Of course, there is a concept of **free multicategory**: Given a set X, and for all $n \in N$, and $x, x_1, \ldots, x_n \in X$, a set $X(x_1, \ldots, x_n; x)$, we get a multicategory X' with $X_0' = X_0$, and $X'(x_1, \ldots, x_n; x)$ given by formal compositions of elements of the $X(y_1, \ldots, y_m; y)$.

A **bicategory** consists of a class $\mathcal{B}_0$ of 0-cells, or objects; For each $A, B \in \mathcal{B}_0$, a category $\mathcal{B}(A, B)$ of 1-cells (objects) and 2-cells (morphisms); for each $A, B, C \in \mathcal{B}_0$, a functor $\mathcal{B}(B, C) \times \mathcal{B}(A, B) \to \mathcal{B}(A, C)$ written $(g, f) \mapsto g \circ f$ on 1-cells and $(\delta, \gamma) \mapsto \delta * \gamma$ on 2-cells; For each $A \in \mathcal{B}_0$ an object $1_A \in \mathcal{B}(A, A)$; isomorphisms representing associativity and identity axioms (e.g. $f \circ 1_A \cong f \in \mathcal{B}(A, B)$), natural in their arguments, satisfying pentagon and triangle axioms.

The collection of categories Cat forms a bicategory. In analogy, we define a monad in a bicategory to be an object A, together with a 1-cell $t : A \to A$ and 2-cells $\mu : t \circ t \to t$ and $\eta : 1_A \to t$ satisfying a couple of commutativity axioms (those of 1.1.3 in [**Lei03**]).

10. Operads

10.1. Definitions. An **operad** is a multicategory with only one object. More explicitly, an operad has a set $P(k)$ for every $k \in \mathbb{N}$, whose elements can be thought of as k-ary operations. It also has, for all $n, k_1, \ldots, k_n \in \mathbb{N}$, a *composition* function

$$P(n) \times P(k_1) \times \cdots \times P(k_n) \to P(k_1 + \cdots + k_n)$$

and an element $1 = 1_P \in P(1)$ called the **identity**, satisfying

$$\theta \circ (1, 1, \ldots, 1) = \theta = \theta \circ 1$$

for all θ, and

$$\theta \circ (\theta_1 \circ (\theta_{1,1}, \ldots, \theta_{1,k_1}), \ldots, \theta_n \circ (\theta_{n,1}, \ldots, \theta_{n,k_n})) = (\theta \circ (\theta_1, \ldots, \theta_n)) \circ (\theta_{1,1}, \ldots, \theta_{n,k_n})$$

for all $\theta \in P(n)$, $\theta_1 \in P(k_1)$, ..., $\theta_n \in P(k_n)$ and all $\theta_{1,1} \ldots \theta_{n,k_n}$.

A **morphism of operads** is a family

$$f_n : (P(n) \to Q(n))_{n \in \mathbb{N}}$$

of functions, preserving composition and identities.

A **P-algebra** for P an operad, is a set X and, for each n, and $\theta \in P(n)$, a function $\overline{\theta} : X^n \to X$, satisfying the evident axioms (identity is the identity function, the function of a composition is the composition of the functions?).

10.2. Examples. For any vector space V, there is an operad with $P(k) = V^{\otimes k} \to V$.

The terminal operad 1 has $P(n) = \{\star_1\}$ for all n. An algebra for 1 is a set X together with a function $X^n \to X$, denoted as $(x_1, \ldots, x_n) \mapsto (x_1 \cdot \cdots \cdot x_n)$, satisfying

$$((x_{1,1} \cdot \ldots x_{1,k_1}) \cdot \cdots \cdot (x_{n,1} \cdot \cdots \cdot x_{n,k_n})) = (x_{1,1} \cdot \cdots \cdot x_{n,k_n})$$

and

$$x = (x).$$

The category of 1-algebras is the category of monoids.

There exist various sub-operads of 1. For example, the smallest one has $P(1) = \{\star\}$ and $P(n) = \emptyset$ for $n \neq 1$.

Or the operad with $P(0) = \emptyset$ and $P(n) = \{\star_n\}$ for $n > 0$, which has semigroups as its algebras (sets with associative binary operations).

The suboperad with $P(n) = \{\star_n\}$ exactly when $n \leq 1$ has as its algebras the pointed sets.

The **operad of curves** has $P(n) = \{\text{smooth maps } \mathbb{R} \to \mathbb{R}^n\}$.

Given a monad on **Set**, we get a natural operad structure $T(n)_{n\in\mathbb{N}}$, with $T(n)$ the set of words in n variables and composition given by 'substitution'.

Given a monoid M (a category with one object), there is a operad given by $P(n) = M^n$ and composition

$$(\alpha_1, \ldots, \alpha_n) \circ ((\alpha_{1,1}, \ldots, \alpha_{1,k_1}), \ldots, (\alpha_{n,1}, \ldots, \alpha_{n,k_n})).$$

The **Little 2-disks** operad D has

$$D(n) = \{\text{set of non-overlapping disks contained within the unit disk}\},$$

with composition being geometric "substitution". I.e.: scale and move a unit disk and its contained disks to match one of the smaller disks, and replace the smaller disk with the transformed contents of our original unit disk. See also: this image that explains a lot

Given sets $X(n)$ for all $n \in \mathbb{N}$, the **free operad** X' on these is defined exactly by $X(n) \subseteq X'(n)$, $1 \in X'(1)$ and for all $m, n_1, \ldots, n_m \in \mathbb{N}$ and $f \in X(m)$ and $f_i \in X'(n_i)$, we have $f \circ (f_1, \ldots, f_m) \in X'(n_1 + \cdots + n_m)$.

11. T-operads

11.1. Definitions. A category is **cartesian** if it has all pullbacks. A functor is cartesian if it preserves pullbacks. A natural transformation $\alpha : S \to T$ is cartesian if for all $f : A \to B$, the naturality diagram

$$\begin{array}{ccc} SA & \xrightarrow{Sf} & SB \\ \downarrow{\scriptstyle \alpha_A} & & \downarrow{\scriptstyle \alpha_B} \\ TA & \xrightarrow{Tf} & TB \end{array}$$

is a pullback. A monad (T, μ, η) on a category $\mathcal{E}$ is cartesian if the category $\mathcal{E}$, the functor T and the natural transformations μ and η are cartesian.

We can represent (the morphism structure of) an ordinary category using diagrams $C_0 \xleftarrow{\text{domain}} C_1 \xrightarrow{\text{codomain}} C_0$, $C_1 \times_{C_0} C_1 \xrightarrow{\text{composition}} C_1$ and $C_0 \xrightarrow{\text{id}} C_1$ together with some axioms. For a multicategory, we need to slightly modify this, using a functor $T : \mathbf{Set} \to \mathbf{Set}$, $A \mapsto \bigsqcup A^n$, to $TC_0 \xleftarrow{d} C_1 \xrightarrow{c} C_0$ and $C_1 \times_{TC_0} TC_1 \xrightarrow{\circ} C_1$.

Given a cartesian monad (T, μ, η) on a category $\mathcal{E}$, we can define a bicategory $\mathcal{E}_{(T)}$, with the class of 0-cells being $\mathcal{E}_0$, the 1-cells $E \to E'$ being diagrams $TE \xleftarrow{d} M \xrightarrow{c} E'$, 2-cells $(M, d, c) \to (N, q, p)$ are maps $M \to N$ such that the diagram with E, E', M, N commutes. The composite of 1-cells $TE \xleftarrow{d} M \xrightarrow{c} E'$ and $TE \xleftarrow{d'}$

$M' \xrightarrow{c'} E''$ is given by

$$TE \xleftarrow{\mu_E} T^2E \xleftarrow{Td} TM \leftarrow TM \times_{TE'} M' \to M' \xrightarrow{c'} E''$$

in which the coproduct in the middle is formed using Tc and d. We can define a T-multicategory to be a monad on $\mathcal{E}_{(T)}$. Equivalently, we can define it as an object $C_0 \in \mathcal{E}$, together with a diagram $t : TC_0 \xleftarrow{d} C_1 \xrightarrow{c} C_0$ and maps $C_1 \circ C_1 := TC_1 \times_{TC_0} C_1 \xrightarrow{\circ} C_1$ and $C_0 \xrightarrow{id} C_1$ satisfying associativity and identity axioms.

A T-operad is a T-multicategory such that C_0 is the terminal object of $\mathcal{E}$. Equivalently, it is an object over $T1$, (so we have a morphism $P \to T1$), together with maps $P \times_{T1} TP \to P$ and $1 \xrightarrow{id} P$, both over $T1$, satisfying associativity and identity axioms.

11.2. Examples. For T the identity monad on **Set**, a T-operad is exactly a monoid (or an operad with only unary functions) (since there is always a unique map to $\{1\}$).

If $\mathcal{E}$ is **Set**, the terminal object 1 will always be $\{1\}$.

For the free monoid monad $TA = \bigsqcup A^n$, the T-operads are precisely the operads that we defined before.

For the monad $TA = 1 + A$, we can view TA as a subset of the free monoid on A, and this gives an operad with 0-ary and 1-ary functions. The 1-ary arrows form a monoid, and the 0-ary arrows are a set, with an action of the monoid.

12. Cartesian Operads

12.1. Theory. Using Towards a doctrine of operads.

NLab uses notation: **Fin** for what we would call a standard skeleton of finite sets (i.e. the category of finite sets $\{0, \dots, n-1\}$ and maps between them). A^B denotes all morphisms/functors $B \to A$. I.e., the class of functors $\mathbf{Fin} \to \mathbf{Set}$ is denoted $\mathbf{Set}^{\mathbf{Fin}}$.

Take $I = \mathbf{Fin}\,(1, -) : \mathbf{Set}^{\mathbf{Fin}} = \mathbf{Fin} \to \mathbf{Set}$.

Let $[\mathbf{Set}^{\mathbf{Fin}}, \mathbf{Set}^{\mathbf{Fin}}]$ be the category of finite-product-preserving, cocontinuous endofunctors on $\mathbf{Set}^{\mathbf{Fin}}$. The map $Ev_I : [\mathbf{Set}^{\mathbf{Fin}}, \mathbf{Set}^{\mathbf{Fin}}] \to \mathbf{Set}^{\mathbf{Fin}}$, given by $F \mapsto F(I)$ is an equivalence. $[\mathbf{Set}^{\mathbf{Fin}}, \mathbf{Set}^{\mathbf{Fin}}]$ has a monoidal product $\odot$ given by endofunctor composition, and we can transfer this to $\mathbf{Set}^{\mathbf{Fin}}$.

Concretely, we have $F \odot G = \int^{n \in \mathbf{Fin}} F(n) G^n$.

12.2. Cartesian operads. A **cartesian operad** is a monoid in this monoidal category $(\mathbf{Set}^{\mathbf{Fin}}, \odot, I)$. I.e., it is a triple (M, μ, η), with $M \in \mathbf{Set}^{\mathbf{Fin}}$, $\mu : M \odot M \to M$ and $\eta : I \to M$.

More concretely, this is a functor $M : \mathbf{Fin} \to \mathbf{Set}$, together with maps

$$m_{n,k} : M(n) \times M(k)^n \to M(k),$$

natural in k and dinatural in n, and an element $e \in M(1)$.

Dinaturality in n means the following. Fix $k \in \mathbf{Fin}$. We have the functors $\mathbf{Fin}^{\mathrm{op}} \times \mathbf{Fin} \to \mathbf{Set}$ given by

$$F : (n, n') \mapsto M(n') \times M(k)^n \quad \text{and} \quad G : (n, n') \mapsto M(k).$$

For all $n \in \mathbf{Fin}_0$, we have a morphism

$$\bullet : F(n, n) = M(n) \times M(k)^n \to M(k) = G(n, n).$$

Naturality means that for all $a : n \to n'$,

$$G(n,a) \circ \bullet \circ F(a,n) = F(a,n') \circ \bullet \circ G(n',a).$$

i.e., for all $f \in M(n)$, $g_1, \dots, g_{n'} \in M(k)$,

$$f \bullet (g_{a(1)}, \dots, g_{a(n)}) = M(a)(f) \bullet (g_1, \dots, g_{n'}).$$

Now, if we have, in **Fin**, a decomposition $k = k_1 + \cdots + k_n$, and we have inclusion maps $i_j : k_j \hookrightarrow k$, then we have

$$M(n) \times M(k_1) \times \cdots \times M(k_n) \xrightarrow{1 \times M(i_1) \times \cdots \times M(i_n)} M(n) \times M(k)^n \xrightarrow{m_{n,k}} M(k),$$

which gives an operad structure.

12.3. Clones. In other parts of mathematics, a cartesian operad is called a **clone**. An abstract clone consists of sets $M(n)$ for all $n \in \mathbb{N}$, for all $n, k \in \mathbb{N}$ a function $\bullet : M(n) \times M(k)^n \to M(k)$ and for each $1 \leq i \leq n \in \mathbb{N}$, an element $\pi_{i,n} \in M(n)$ such that for $f \in M(i)$, $g_1, \dots, g_i \in M(j)$ and $h_1, \dots, h_j \in M(k)$,

$$f \bullet (g_1 \bullet (h_1, \dots, h_j), \dots, g_i \bullet (h_1 \dots, h_j)) = (f \bullet (g_1, \dots, g_i)) \bullet (h_1, \dots, h_j),$$

for $f_1, \dots, f_n \in M(k)$,

$$\pi_{i,n} \bullet (f_1, \dots, f_n) = f_i,$$

and for $f \in M(n)$,

$$f \bullet (\pi_{1,n}, \dots, \pi_{n,n}) = f.$$

(It is claimed that this automatically gives naturality)

Naturality means for all $a \in \mathbf{Fin}\,(n, n')$, for all $f \in M(n)$, $g_1, \dots, g_{n'} \in M(k)$,

$$f \bullet (g_{a(1)}, \dots, g_{a(n)}) = (f \bullet (\pi_{a(1),n'}, \dots, \pi_{a(n),n'})) \bullet (g_1, \dots, g_{n'}).$$

However, by associativity and since $\pi_{i,n} \bullet (f_1, \dots, f_n)$, we have

$$\begin{aligned}
&(f \bullet (\pi_{a(1),n'}, \dots, \pi_{a(n),n'})) \bullet (g_1, \dots, g_{n'}) \\
&= (f \bullet (\pi_{a(1),n'} \bullet (g_1, \dots, g_{n'}), \dots, \pi_{a(n),n'} \bullet (g_1, \dots, g_{n'}))) \\
&= f \bullet (g_{a(1)}, \dots, g_{a(n)}).
\end{aligned}$$

Naturality in k is as follows. Fix $n \in \mathbf{Fin}_0$. We have functors $F, G : \mathbf{Fin} \to \mathbf{Set}$, given by

$$F : k \mapsto M(n) \times M(k)^n \quad \text{and} \quad G : k \mapsto M(k).$$

For $a : k \to k'$, we must have

$$G(a) \circ \bullet = \bullet \circ F(a).$$

That is, for all $f \in M(n)$, $g_1, \dots, g_n \in M(k)$,

$$M(a)(f \bullet (g_1, \dots, g_n)) = f \bullet (M(a)(g_1), \dots, M(a)(g_n)).$$

Now, $M(a)$ is given by

$$M(a)(f) = f \bullet (\pi_{a(1),k'}, \dots, \pi_{a(k),k'}).$$

Therefore, we have

$$\begin{aligned}
&M(a)(f \bullet (g_1, \dots, g_n)) \\
&= (f \bullet (g_1, \dots, g_n)) \bullet (\pi_{a(1),k'}, \dots, \pi_{a(k),k'}) \\
&= f \bullet (g_1 \bullet (\pi_{a(1),k'}, \dots, \pi_{a(k),k'}), \dots, g_n \bullet (\pi_{a(1),k'}, \dots, \pi_{a(k),k'})) \\
&= f \bullet (M(a)(g_1), \dots, M(a)(g_n)).
\end{aligned}$$

So the associativity and projection axioms ensure naturality.

Week 09

13. Adjunctions

For three sets X, Y, Z, we have a bijection

$$\mathbf{Set}\left(X \times Y, Z\right) \cong \mathbf{Set}\left(X, Y \to Z\right)$$

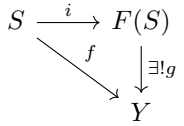
that is natural in X and Z. This is an example of a general notion:

For functors $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{D} \to \mathcal{C}$, F is a **left adjoint** for G if there is an isomorphism $\mathcal{C}\left(FX, Y\right) \cong \mathcal{D}\left(X, GY\right)$ that is dinatural in X and Y.

13.1. Examples. A special case is for $G : \mathcal{C} \to \mathbf{Set}$ the forgetful functor: if we have a left adjoint $F : \mathbf{Set} \to \mathcal{C}$ to G, we call $F(X)$ the **free** $\mathcal{C}$ of X (for example: free group, free monoid, free category, etc.). That means that $\mathcal{C}\left(F(X), Y\right) \cong \mathbf{Set}\left(X, G(Y)\right)$, or in other words:

Adjunctions can be composed: If F and G both forget part of the structure of an object, then we can construct the free object 'piecewise' using the composition of the left adjoints.

Let $G : \mathbf{Group} \to \mathbf{Set}$ be the forgetful functor and $F : \mathbf{Set} \to \mathbf{Group}$ be the free group functor. For any set S, we have an inclusion $i : S \hookrightarrow FS$. Now, given any group Y and any morphism of sets from S to Y (formally: $f \in \mathbf{Set}\left(S, GY\right)$). Then the fact that $\mathbf{Set}\left(FS, Y\right) \cong \mathbf{Set}\left(S, GY\right)$ is expressed in the fact that f factors uniquely as $g \circ i$.



In topology, the forgetful functor from topological spaces to sets has a left adjoint that turns a set S into a space $F(S)$ with the discrete topology, since it has the smallest topology on S that still can factor all morphisms $S \to [0, 1]$ for example.

In algebra, the inclusion functor from the category of monoids (sets with an associative operation and a unit) to the category of semigroups (sets with an associative operation) has a left adjoint that sends the semigroup S to the monoid $S \sqcup \{\star\}$ and gives it operations such that $\star$ is the unit.

In algebra, the inclusion functor from the category of rings to the category of rngs (rings but without an identity) has a left adjoint that sends the rng R to $R \times \mathbb{Z}$ and defines $(r, x)(s, y) = (rs + xs + ry, xy)$.

For $\mathbf{Dom_m}$ the category of integral domains with injective morphisms. Then the forgetful functor $\mathbf{Fields} \to \mathbf{Dom_m}$ has a left adjoint that sends a domain to its field of fractions.

For $\rho : R \to S$ a morphism of rings, we can view every S-module as an R-module: we have a functor $S\text{−}\mathbf{Mod} \to R\text{−}\mathbf{Mod}$. This has a left adjoint, $M \mapsto M \otimes_R S$.

14. Operads and cartesian operads

14.1. Recap operads and cartesian operads. In this section, we will represent a cartesian operad C by a clone:

- For all $n \in \mathbb{N}$, a set $C(n)$;
- For all $1 \leq i \leq n$ projections $\pi_{n,i} \in P(n)$;
- For all m, n, a function $\rho_{m,n} : C(m) \times C(n)^m \to C(n)$.

such that

- For $f \in C(l)$, $g_1, \ldots, g_l \in C(m)$ and $h_1, \ldots, h_m \in C(n)$,

$$\rho(f, (\rho(g_1, (h_1, \ldots, h_m)), \ldots, \rho(g_l, (h_1, \ldots, h_m)))) = \rho(\rho(f, (g_1, \ldots, g_l)), (h_1, \ldots, h_l)).$$

- for $f_1, \ldots, f_n \in M(n)$ and $1 \leq i \leq n$, $\rho(\pi_{i,n}, (f_1, \ldots, f_n)) = f_i$;
- For $f \in M(n)$, $\rho(f, (\pi_{1,n}, \ldots, \pi_{n,n})) = f$.

and a morphism of cartesian operads $\varphi : C \to C'$ by componentwise functions $\varphi_n : C(n) \to C'(n)$ such that

- For all $1 \leq i \leq n$, $\varphi_n(\pi_{n,i}) = \pi'_{n,i}$;
- For all m, n, $f \in C(m)$, $g_1, \ldots, g_m \in C(n)$

$$\rho(\varphi(f), (\varphi(g_1), \ldots, \varphi(g_m))) = \varphi(\rho(f, (g_1, \ldots, g_m))).$$

We will represent an operad P by

- For all $n \in \mathbb{N}$, a set $P(n)$;
- An element $e \in P(1)$;
- For all $m, n_1, \ldots, n_m \in \mathbb{N}$, a function

  $$\gamma_{m,n_1,\ldots,n_m} : C(m) \times C(n_1) \times \cdots \times C(n_m) \to C(n_1 + \cdots + n_m)$$

  (we will usually just call this γ).

such that

- For all $f \in P(n)$,

$$\gamma(f, (1, \ldots, 1)) = f = \gamma(1, f);$$

- For all $f \in P(l)$, $g_1 \in P(m_1)$, …, $g_l \in P(m_l)$ and all $h_{1,1} \in P(n_{1,1})$, …, $h_{l,m_l} \in P(n_{l,m_l})$,

$$f \circ (g_1 \circ (h_{1,1}, \ldots, h_{1,m_1}), \ldots, g_l \circ (h_{l,1}, \ldots, h_{l,m_l})) = (f \circ (g_1, \ldots, g_l)) \circ (h_{1,1}, \ldots, h_{l,m_l}).$$

and a morphism of operads $\psi : P \to P'$ by componentwise functions $\psi_n : P(n) \to P'(n)$ such that

- $\psi_1(e) = e'$;
- For all $f \in P(m)$, $g_1 \in P(n_1)$, …, $g_m \in P(n_m)$,

$$\gamma(\psi(f), (\psi(g_1), \ldots, \psi(g_m)))\psi(\gamma(f, (g_1, \ldots, g_m))).$$

14.2. Free operad? Now, for all $1 \leq i \leq m$, we have an injection $j_i : C(n_i) \to C(n_1 + \cdots + n_m)$. Intuitively, it maps terms in a context with n_i variables to terms in a context with $n_1 + \cdots + n_m$ variables, by mapping variable $1 \leq k \leq n_1$ to variable $n_1 + \cdots + n_{i-1} + k$. This gives a morphism

$$\gamma : C(m) \times C(n_1) \times \cdots \times C(n_m) \xrightarrow{id \times j_1 \times \cdots \times j_m} C(m) \times C(n_1 + \cdots + n_m)^m \xrightarrow{\rho} C(n_1 + \cdots + n_m),$$

which gives an operad structure on the same sets. This gives a functor from cartesian operads to operads.

Then, the question arises whether this functor has a left adjoint.

Still open

15. Cartesian multicategory

Yet another way to think of a cartesian operad is as a one-object cartesian multicategory. A **cartesian multicategory** is a multicategory with an S_n-action on the hom-sets (in other words: we can permute the arguments of the morphsims) and duplication/diagonal/contraction operations

$$\mathrm{Hom}(c_1, \ldots, c_k, c_k, \ldots, c_n; c) \to \mathrm{Hom}(c_1, \ldots, c_k, \ldots, c_n; c)$$

and deletion/projection/weakening operations

$$\mathrm{Hom}(c_1, \ldots, c_k, \ldots, c_n; c) \to \mathrm{Hom}(c_1, \ldots, c_{k-1}, c_{k+1}, \ldots, c_n; c)$$

16. The paper

A summary of the results in the paper:

(1) A definition of the category of algebraic theories (clones).
(2) A definition of the category of T-algebras (for T an algebraic theory), the pullback of an algebra, the free algebra, and some properties of these.
(3) A definition of a presheaf (like an algebra, but with a right action instead of a left one) and some properties of this.
(4) A sidenote about the more classical approach to algebra.
(5) A definition for a λ-theory and some properties of its constituents.
(6) A definition for an interpretation of the λ-calculus in a λ-theory, and some properties.
(7) The notion that the algebraic theory Λ of all terms of the λ-calculus is the initial λ-theory.
(8) The notion that we can add constants to a theory to make sure that it 'has enough points'.
(9) A (new) proof that any λ-theory is isomorphic to the endomorphism λ-theory of some object.
(10) A section that concludes an equivalence between the presheaf categories of a Lawvere theory, a λ-theory and the category of retracts.
(11) The definition of Λ-algebra and a couple of its properties. In particular, a functor from λ-theories to Λ-algebras.
(12) An equivalene between Λ-algebras and their presheaves.
(13) A characterisation of the function space U^U for $U \in PA$ the universal.
(14) An equivalence of categories between the category of Λ-algebras and λ-theories.
(15) The Fundamental Theorem of the λ-calculus: there is an adjoint equivalence between λ-theories and Λ-algebras.

(16) A remark that this is much harder without the category theoretic framework.

Week 11

17. A Formalization of Operads in Coq

A paper was uploaded to the arXiv (see [**FTBF23**]). Unfortunately, this paper did not contain any code, which makes it harder to compare their technique to mine. The paper is about the formalization of a symmetric multicategory (kind of. They do not require $(\tau\sigma)f = \tau(\sigma f)$ for $\sigma, \tau \in S_n$ and $f \in \mathcal{C}\,(a_1, \ldots, a_n; a)$)

The operad data presented in this paper is a bit different (but equivalent) from the data that I have seen so far. Notably: the composition is on one element at a time:

$$\mathcal{C}\,(a_1, \ldots, a_n; a) \times \mathcal{C}\,(b_1, \ldots, b_m; a_i) \to \mathcal{C}\,(a_1, \ldots a_{i-1}, b_1, \ldots, b_m, a_{i+1}, \ldots, a_n; a)\,,$$

in which $a_1, \ldots a_{i-1}, b_1, \ldots, b_m, a_{i+1}, \ldots, a_n$ is denoted $\underline{a} \bullet_i \underline{b}$.

I expect the axioms to become a bit more complicated this way. For example, now there are two associativity axioms, and even to state them we need a proof that $(\underline{c} \bullet_i \underline{a}) \bullet_{l-1+j} \underline{b} = (\underline{c} \bullet_j \underline{b}) \bullet_i \underline{a}$.

Also, their signature of Hom is **Type** $\to$ **List Type** $\to$ **Type**. The advantage of using **List Type** instead of $\sum_{n \in \mathbb{N}}(\texttt{stn}\ n \to \textbf{Type})$ is that it sounds simpler. The disadvantage is that, for example, to say something about c_i, one needs a function to fetch the ith element of $\underline{c}$ and a proof that i is less than the length of $\underline{c}$.

To get elements in the right space, they add a lot of typecasting functions. For example, the function $\mathcal{C}_{assoc} : \mathcal{O}((\underline{c} \bullet_i \underline{a}) \bullet_{l-1+j} \underline{b}; d) \to \mathcal{O}((\underline{c} \bullet_j \underline{b}) \bullet_i \underline{a}; d)$ I believe we call such a function a 'transport function'. The paper claims that if $A = B$, a typecast function will be the identity. The way that it is currently stated, it is not compatible with univalence, I believe.

I feel like the paper lacks a section explaining the choices that were made and evaluating their results.

Week 17

18. The applicability of algebraic theories

It turns out that it is very well possible to characterize a family of algebraic constructs using algebraic theories. For example, if we take the algebraic theory $\mathcal{T}$ with $\mathcal{T}(n)$ the monoids on n generators, there is an equivalence (at least of types, maybe also of categories) between $\mathcal{T}$-algebras and monoids. We can do the same for groups, rings, ring modules, ring algebras etc. The action of certain elements corresponds then to group operation, the ring multiplication, the ring action, etc:

$$a \cdot b := \alpha_2(x_1 \cdot x_2, (a, b)).$$

The laws that these operations must obey can also be deduced from the action, by pulling back the equation to the free object. For example, associativity:

$$\begin{aligned} a \cdot (b \cdot c) &= \alpha_2(x_1 \cdot x_2, (a, \alpha_2(x_1 \cdot x_2, (b, c)))) \\ &= \alpha_2(x_1 \cdot x_2, (\alpha_3(x_1, (a, b, c)), \alpha_2(x_1 \cdot x_2, (\alpha_3(x_2, (a, b, c)), \alpha_3(x_3, (a, b, c)))))) \\ &= \alpha_3(x_1 \cdot (x_2 \cdot x_3), (a, b, c)) \\ &\dots \\ &= (a \cdot b) \cdot c. \end{aligned}$$

However, not all algebraic objects can be expressed as algebras for some algebraic theory in this way. Consider, for example, fields. If we have some algebraic theory $\mathcal{T}$ that has the fields as its algebras, we would expect some element $x_1^{-1} = \frac{1}{x_1} \in \mathcal{T}(1)$ that expresses the multiplicative inverse. However, there is no good way to give a value to $\alpha_1(x_1^{-1}, (0))$, since 0 does not have a multiplicative inverse.

In general, given a category $\mathcal{C}$, a functor $F : \mathcal{C} \to \mathbf{Set}$ with a left adjoint $G : \mathbf{Set} \to \mathcal{C}$. Then we can set

$$T(n) = F(G(\{1, \dots, n\})).$$

Given $c \in \mathcal{C}_0$, we have a natural bijection $\varphi : \mathbf{Set}\left(\{1, \dots, n\}, F(c)\right) \xrightarrow{\sim} \mathcal{C}\left(G(\{1, \dots, n\}), c\right)$. Then, for $f \in T(m)$, $g : \{1, \dots, m\} \to T(n)$, we can set $f \bullet g := F(\varphi(g))(f)$. We also take $\pi_{n,i} = \varphi^{-1}(\mathrm{id}_{G(\{1,\dots,n\})})(i)$. Then, given any $A \in \mathcal{C}_0$, we can make $F(A)$ into an algebra by setting $\alpha_n(f, a) = F(\varphi(a))(f)$. Also, given a morphism $m \in \mathcal{C}\left(A, B\right)$, we have a function $F(m) \in \mathbf{Set}\left(F(A), F(B)\right)$. are equal. Naturality gives for all $a : \{1, \dots, n\} \to F(A)$ that

$$m \circ \varphi(a) = \varphi(F(m) \circ a),$$

so for $f : F(G(\{1, \dots, n\}))$,

$$F(m)(\alpha_2(f, a)) = F(m \circ \varphi(a))(f) = F(\varphi(F(m) \circ a))(f) = \alpha_2(f, F(m) \circ a)$$

and $F(m)$ is a morphism of algebras. Of course, F respects composition and identities, so we have a functor from objects of $\mathcal{C}$ to T-algebras.

However, this functor is not necessarily an equivalence of categories. For example, take $\mathcal{C} := \mathbf{Set} \times \mathbf{Set}$, with $F(A,B) = A$, $G(A) = (A, \emptyset)$ and $\varphi(f) = (f, \iota_\emptyset)$. Then $T(n) = \{1, \ldots, n\}$. Take $(A,B) \in \mathbf{Set} \times \mathbf{Set}_0$. Then $F(A,B) = A$. Also, for all n and all $a : \{1,\ldots,n\} \to A$, $\varphi(a) = (a, \iota_\emptyset)$ and $F(\varphi(a)) = a$ again. Therefore, $\alpha_n(f,a) = a(f)$, regardless of B, so the functor is not an equivalence of categories.

19. Using displayed categories

Displayed categories are a way to build categories in a layered fashion. I replaced my old definitions for the objects in my library and their categories by the displayed categories version, shaping the definitions of my objects and morphisms in such a way that they are definitionally equal to the objects and morphisms in the (total) displayed categories.

It turns out that coq has a hard time doing coercions between categories, since there are a lot of implicit coercions between a category and the type of its objects. So for a displayed category $\mathcal{C}'$ over $\mathcal{C}$, and a displayed category $\mathcal{C}''$ over $\mathcal{C}'$, if we have coercions $\mathcal{C}'' \to \mathcal{C}'$ and $\mathcal{C}' \to \mathcal{C}$, if we have an object of type $\mathcal{C}''$ (which is actually ${\mathcal{C}''}_0$) and we need an object of type $\mathcal{C}$ (actually $\mathcal{C}_0$), then coq will need to make the following chain of coercions:

$$ {\mathcal{C}''}_0 \to \mathcal{C}'' \to \mathcal{C}' \to \mathcal{C} \to \mathcal{C}_0 $$

and this does not go very well. In this case, it worked very well to define separate (definitionally equal) datatypes for the objects and morphisms in each category, define coercions between those, and cast objects in our categories to these analogous datatypes in places where we need these coercions.

Week 18

LEMMA 1. *The lambda calculus Λ is the initial λ-theory.*

PROOF. Given a λ-theory $\mathcal{L}$, we need a unique morphism $\varphi : \Lambda \to \mathcal{L}$.

Do structural induction on the lambda terms in $\Lambda(n)$. Send `Var(i)` to $\pi_{n,i}$, `Abs(f)` to $\lambda(\varphi(f))$ and `App(f, g)` to $\rho(\varphi(f)) \bullet (\pi_1, \ldots, \pi_n, g)$.

Show that it preserves $\bullet$, π_i, ρ and λ.

Uniqueness: Again, structural induction on the lambda terms. Preservation of the π_i gives the image of `Var(i)`. Preservation of λ gives the image of `Abs(f)`. Preservation of ρ and $\bullet$ gives the image of

$$\texttt{App}(\texttt{f}, \texttt{g}) = \rho(f) \bullet (\pi_1, \ldots, \pi_n, g).$$

□

Given a λ-theory $\mathcal{L}$, $\mathcal{L}(0)$ is a Λ-algebra (by extension of scalars). This gives a functor.

LEMMA 2. *Given a Λ-algebra A, we can form the set $\mathcal{U}_A$.*

PROOF. □

LEMMA 3. *$\mathcal{U}_A$ is a λ-theory.*

LEMMA 4. *$\mathcal{U}_A$ is functorial in A.*

PROOF. □

These functors form an adjoint equivalence: We have natural isomorphisms $\mathcal{U}_{\mathcal{L}(0)} \cong \mathcal{L}$ and $\mathcal{U}_A(0) \cong A$.

Week 33

20. Coends

In a category $\mathcal{D}$ with equalizers and products, and given a functor $F : \mathcal{C}^{\mathrm{op}} \times \mathcal{C} \to \mathcal{D}$, we can define the end $\int_{c:\mathcal{C}_0} F(c,c)$ to be the equalizer of

$$\prod_{c:\mathcal{C}_0} F(c,c) \rightrightarrows \prod_{\mathcal{C}(c,c')} F(c,c').$$

For $\mathcal{D} = \mathbf{Set}$, an element of this gives, for every element $c : \mathcal{C}_0$, an element $f_c : F(c,c)$, such that for all morphisms $\alpha : \mathcal{C}(c,c')$, we have

$$\#F(c,\alpha)(f_c) =_{F(c,c')} \#F(\alpha,c')(f_{c'}).$$

If we have two functors $G, G' : \mathcal{C} \to \mathcal{C}'$, take $F(c,c') = \mathcal{D}'(G(c), G'(c'))$. An element of $\int_c \mathcal{D}'(G(c), G'(c'))$ gives, for all $c : \mathcal{C}_0$ an element $f_c : \mathbf{Set}(G(c), G(c'))$ such that for all morphisms $\alpha : \mathcal{C}(c,c')$, we have

$$f_c \cdot \#G'(\alpha) = \#F(c,\alpha)(f_c) = \#F(\alpha,c')(f_{c'}) = \#G(\alpha) \cdot f_{c'},$$

so we have a commutative diagram

$$\begin{array}{ccc} G(c) & \xrightarrow{\#G(\alpha)} & G(c') \\ \downarrow{\scriptstyle f_c} & & \downarrow{\scriptstyle f_{c'}} \\ G'(c) & \xrightarrow{\#G'(\alpha)} & G'(c) \end{array}$$

Which is exactly the diagram of a natural transformation.

On the other hand, if $\mathcal{D}$ has coproducts, we can define the coend $\int^{c:\mathcal{C}_0} F(c,c)$ to be the coequalizer of

$$\coprod_{\mathcal{C}(c,c')} F(c,c') \rightrightarrows \coprod_{c:\mathcal{C}_0} F(c,c).$$

For $\mathcal{D} = \mathbf{Set}$, an element of this is an element $c : \mathcal{C}_0$, together with an element of $F(c,c)$. For $c, c' : \mathcal{C}_0$, $f : F(c',c)$ and $\alpha : \mathcal{C}(c,c')$, we identify

$$(c, \#F(\alpha,c)(f))$$

with

$$(c', \#F(c',\alpha)(f)).$$

For $F(c,c') = \mathcal{C}'(G(c), G'(c))$, an element of $\int^{c:\mathcal{C}_0}$ is an element $c : \mathcal{C}_0$ with a morphism $f : \mathcal{C}'(G(c), G'(c))$ and for a morphism $\alpha : \mathcal{C}(c,c')$ and morphism $f : \mathcal{C}'(G(c'), G'(c))$, we set

$$(c, \#G(\alpha) \cdot f) = (c, \#F(\alpha,c)(f)) = (c', \#F(c',\alpha)(f)) = (c', f \cdot \#G'(\alpha)).$$

20.1. Wedges. A wedge $e : w \to F$ is an object w with, for every object $c : \mathcal{C}_0$ a morphism $e_c : w \to F(c, c)$ and for every morphism $\alpha : c \to c'$ a commutative square

$$\begin{array}{ccc} w & \xrightarrow{e_{c'}} & F(c', c') \\ \downarrow{\scriptstyle e_c} & & \downarrow{\scriptstyle \#F(\alpha, c')} \\ F(c, c) & \xrightarrow{\#F(c,\alpha)} & F(c, c') \end{array}$$

The end $e : \int_{c:\mathcal{C}_0} F(c, c) \to F$ is the universal wedge. I.e., given a wedge $e' : w' \to F$, e' factors through e via a unique map $w' \to \int_{c:\mathcal{C}_0} F(c, c)$.

Conversely, a cowedge $e : F \to w$ is an object w with, for every object $c : \mathcal{C}_0$ a morphism $e_c : F(c, c) \to w$ and for every morphism $\alpha : c \to c'$ a commutative square

$$\begin{array}{ccc} F(c', c) & \xrightarrow{\#F(c',\alpha)} & F(c', c') \\ \downarrow{\scriptstyle \#F(\alpha, c)} & & \downarrow{\scriptstyle e_{c'}} \\ F(c, c) & \xrightarrow{e_c} & w \end{array}$$

The coend $e : F \to \int^{c:\mathcal{C}_0} F(c, c)$ is the universal cowedge. I.e., given a cowedge $e' : F \to w'$, e' factors through e via a unique map $\int_{c:\mathcal{C}_0} F(c, c) \to w'$.

20.2. The coproduct of algebras. Now, to the example that we are interested in

$$A + B = \int^{m,n:F_0} \mathrm{Alg}_T(T(m), A) \times \mathrm{Alg}_T(T(n), B) \times T(m + n).$$

We can identify $\mathrm{Alg}_T(T(m), A) \cong A^m$:

- Send $f : \mathrm{Alg}_T(T(m), A)$ to $(f\pi_1, \ldots, f\pi_m) : A^m$;
- Send $(a_1, \ldots, a_m) : A^m$ to $x \mapsto x \bullet (a_1, \ldots, a_m) : A$.

Then this becomes

$$A + B = \int^{m,n:F_0} A^m \times B^n \times T(m + n).$$

An element of this set is a pair

$$((m, n), ((a_1, \ldots, a_m), (b_1, \ldots, b_n), f))$$

with $m, n : F_0$, $a_i : A$, $b_i : B$ and $f : T(m + n)$.

If we have $m, m', n, n' : F_0$, $\alpha : F\,(m, m')$, $\beta : F\,(n, n')$, and an element

$$((a_i)_i, (b_i)_i, f) : A^{m'} \times B^{n'} \times T(m + n),$$

we identify

$$((m, n), ((a_{\alpha(i)})_i, (b_{\beta(i)})_i, f)) = ((m', n'), ((a_i)_i, (b_i)_i, \#T(\alpha + \beta)(f)))$$

Now, we can define the action

$$f \bullet (((m, m'), ((a_{ji})_i, (b_{ji})_i, g_j))_j) := ((m, m'), (f \bullet (a_{ji})_j)_i, (f \bullet (b_{ji})_j)_i, f \bullet (g_j)_j).$$

This is associative and projects components because the action on A and B is associative and projects components.

20.2.1. *The universal property.* Now, we would just need to show that for all $C : \mathrm{Alg}_{T0}$ and all $f_A : A \to C$ and $f_B : B \to C$, there exists a unique morphism $g : A + B \to C$.

Since $A+B$ is a coend, it would suffice to give C a $A^m \times B^n \times T(m+n)$-cowedge structure. That is, we need, for all $m, n : F_0$, a morphism $A^m \times B^n \times T(m+n) \to C$. Define

$$f_{A+B}((a_1, \ldots, a_m), (b_1, \ldots, b_n), f) := f \bullet (f_A(a_1), \ldots, f_A(a_m), f_B(b_1), \ldots, f_B(b_n)).$$

We must show that given $x : T(l)$ and given, for all $1 \leq i \leq l$, $(a_i, b_i, f_i) : A^m \times B^n \times T(m+n)$, we have

$$x \bullet (f_{A+B}(a_i, b_i, f_i))_i = f_{A+B}((x \bullet (a_i, b_i, f_i))_i).$$

But

$$\begin{aligned}
&x \bullet (f_i \bullet ((f_A(a_{ij}))_j, (f_B(b_{ij}))_j))_i \\
&= x \bullet (f_{A+B}(a_i, b_i, f_i))_i \\
&= f_{A+B}((x \bullet (a_i, b_i, f_i))_i) \\
&= f_{A+B}((x \bullet (a_{ij})_i)_j, (x \bullet (b_{ij})_i)_j, x \bullet f) \\
&= (x \bullet f) \bullet ((f_A(x \bullet (a_{ij})_i))_j, (f_B(x \bullet (b_{ij})_i))_j) \\
&= (x \bullet f) \bullet ((x \bullet (f_A(a_{ij}))_i)_j, (x \bullet (f_B(b_{ij}))_i)_j)
\end{aligned}$$

Week 34

21. Algebraic Theories as categories

Hyland cites [**ARV11**], which also introduces algebraic theories. There is a correspondence between Hyland's algebraic theories, and "one-sorted algebraic theories" in the book.

The book defines the category $\mathcal{N}$, which is the full subcategory of $\mathbf{Set}^{\mathrm{op}}$, with objects $n = \{0, 1, \ldots, n-1\}$ for all $n \in \mathbb{N}$. This category presents n as 1^n, with projections $\pi_{n,i} : n \to 1$.

21.1. Algebraic Theories. The book defines a one-sorted algebraic theory to be a category with finite products, whose objects are the natural numbers, together with a functor $T : \mathcal{N} \to \mathcal{T}$ that is the identity on objects.

Given a one-sorted algebraic theory $(\mathcal{T}, T)$, we make Hyland's algebraic theory $\mathcal{T}'$ as follows: Take $\mathcal{T}'(n) = \mathcal{T}(n, 1)$. We have $\mathcal{T}'(n)^m = \mathcal{T}(n, 1)^m = \mathcal{T}(n, m)$. Then composition gives

$$\bullet : \mathcal{T}'(m) \times \mathcal{T}'(n)^m = \mathcal{T}(m, 1) \times \mathcal{T}(n, m) \to \mathcal{T}(n, 1) = \mathcal{T}'(n).$$

The projections are given by $\pi'_{n,i} = \#T(\pi_{n,i}) : \mathcal{T}(n, 1) = \mathcal{T}'(n)$.

Conversely, given Hyland's algebraic theory $\mathcal{T}'$, we make a one-sorted algebraic theory $\mathcal{T}$ as follows. Take $\mathcal{T}_0 = \mathbb{N}$ and $\mathcal{T}(m, n) = \mathcal{T}'(m)^n$. Then T sends $f = (\pi_{i_1}, \ldots, \pi_{i_n}) = \mathcal{N}(m, 1)^n = \mathcal{N}(m, n)$ to $(\pi'_{i_1}, \ldots, \pi'_{i_n})$. Composition is then defined as

$$\mathcal{T}(l, m) \times \mathcal{T}(m, 1)^n = \mathcal{T}(l, m) \times \mathcal{T}(m, n) \ni (f, (g_1, \ldots, g_n)) = (f, g) \mapsto (g_1 \bullet f, \ldots, g_n \bullet f) : \mathcal{T}(l, 1)^n = \mathcal{T}(l, n).$$

21.2. Algebraic theory algebras. Given a one-sorted algebraic theory $(\mathcal{T}, T)$ and the corresponding Hyland's algebraic theory $\mathcal{T}'$.

An algebra for $\mathcal{T}$ is a finite-product-preserving functor $A : \mathcal{T} \to \mathbf{Set}$. Given such an algebra, we get an algebra A' for $\mathcal{T}'$ by setting $A' = A(1)$, and for $f : \mathcal{T}'(n) = \mathcal{T}(n, 1)$, we define $f \bullet -$ to be

$$\#A(f) : (A')^n = A(1)^n = A(1^n) = A(n) \to A(1) = A'.$$

Conversely, given an algebra A' for $\mathcal{T}'$, we define the functor A to map n to $(A')^n$ on objects, and to send the morphism $f : \mathcal{T}(m, n) = \mathcal{T}(m, 1)^n = (\mathcal{T}'(m))^n$ to

$$A(m) = (A')^m \ni a \mapsto (f_1 \bullet a, \ldots, f_n \bullet a)(A')^n = A(n).$$

21.3. Pre(- or post)liminaries.

21.3.1. *The Density Theorem.* Given an algebraic theory (can be one-sorted) $\mathcal{T}$, we have a Yoneda embedding $Y_{\mathcal{T}} : \mathcal{T}^{\mathrm{op}} \to [\mathcal{T}, \mathbf{Set}]$, sending X to $\mathcal{T}(X, -)$. Also, given a $\mathcal{T}$-algebra A, we have the category of elements $El\, A$, with the objects being dependent pairs $\sum_{x:\mathcal{T}_0}, Ax$ and morphisms $\mathrm{El}A((X,x),(Y,y))$ being morphisms $f : \mathcal{T}(Y,X)$ with $\#A(f)(y) = x$. Denote $\Phi_A : El\, A \to \mathcal{T}^{\mathrm{op}}$ the projection on the first coordinate. Then A is the colimit of the diagram

$$El\, A \xrightarrow{\Phi_A} \mathcal{T}^{\mathrm{op}} \xrightarrow{Y_{\mathcal{T}}} [\mathcal{T}, \mathbf{Set}].$$

I.e. we have, for all $(x,y) : El\, A$, a map of functors (i.e. natural transformation)

$$Y_{\mathcal{T}}(\Phi_A(x,y)) = Y_{\mathcal{T}}(x) = \mathcal{T}(x,-) \Rightarrow A$$

That is, for all $t : \mathcal{T}$, a map

$$\mathcal{T}(x,t) \to A(t), f \mapsto \#A(f)(y)$$

and for all morphisms $g : \mathcal{T}(t,t')$ and all $f : \mathcal{T}(x,t)$, we have the equality

$$\#A(g)(\#A(f)(y)) = \#A(g \circ f)(y)$$

so the map is natural.

Now, for all $B : [\mathcal{T}, \mathbf{Set}]$, we have

$$\begin{aligned}
[\mathcal{T}, \mathbf{Set}]\left(\underset{(x,y):El\, A}{colim}\, Y_{\mathcal{T}}(\Phi_A(x,y)), B\right) &\cong \lim_{(x,y):El\, A} [\mathcal{T}, \mathbf{Set}]\left(Y_{\mathcal{T}}(\Phi_A(x,y)), B\right) \\
&\cong \lim_{(x,y):El\, A} [\mathcal{T}, \mathbf{Set}]\left(Y_{\mathcal{T}}(x), B\right) \\
&\cong \lim_{(x,y):El\, A} B(x) \\
&\cong [El\, A, \mathbf{Set}]\left(\mathrm{pt}, B\right)
\end{aligned}$$

for pt the constant functor that sends anything to $\{\star\}$. Now, given a natural transformation $\alpha : [El\, A, \mathbf{Set}](\mathrm{pt}, B)$. For all $x : \mathcal{T}_0$, we get a map

$$A(x) \to B(x), \quad y \mapsto \alpha_{(x,y)}(\star).$$

By naturality of α, this is a natural transformation $A \Rightarrow B$. This gives an isomorphism

$$[El\, A, \mathbf{Set}]\left(\mathrm{pt}, B\right) \cong [\mathcal{T}, \mathbf{Set}]\left(A, B\right).$$

Since this holds for all B, we have

$$\begin{aligned}
Y_{[\mathcal{T}, \mathbf{Set}]^{\mathrm{op}}}(A) &= [\mathcal{T}, \mathbf{Set}]\left(A, -\right) \\
&\cong [\mathcal{T}, \mathbf{Set}]\left(\underset{(x,y):El\, A}{colim}\, Y_T(\Phi_A(x,y)), -\right) \\
&= Y_{[\mathcal{T}, \mathbf{Set}]^{\mathrm{op}}}(\underset{(x,y):El\, A}{colim}\, Y_T(\Phi_A(x,y))).
\end{aligned}$$

By the Yoneda Lemma, this implies that $colim_{(x,y):El\, A}\, Y_T(\Phi_A(x,y)) \cong A$.

21.4. An example. First of all, we need an example. Take an algebraic theory (Hyland's kind): $\mathcal{T}'(n) = \{\star, 1, 2, 3, \ldots, n\}$ with $\star \bullet g = \star$ and $i \bullet g = g_i$. Take the two-element set $A' = \{\bot, \top\}$. We can make A' into an $\mathcal{T}'$-algebra by setting $i \bullet a := a_i$ and $\star \bullet a = \top$ (or $\bot$ for that matter, as long as we are consistent).

Translating this to a one-sorted algebraic theory, we get the category $\mathcal{T}$ with $\mathcal{T}_0 = \{0, 1, 2, \ldots\}$ and $\mathcal{T}(m,n) = \{\star, 1, \ldots, m\}^n$. We also get the algebraic theory

(i.e. functor) $A : \mathcal{T} \to \mathbf{Set}$, given on objects by $n \mapsto \{\bot, \top\}^n$ and on morphisms by

$$\mathcal{T}(m,1) \ni f \mapsto a \mapsto \begin{cases} a_i & f = i \\ \top & f = \star \end{cases}$$

and extending linearly for $f \in \mathcal{T}(m,n) = \mathcal{T}(m,1)^n$. The category El$A$ consists of objects (n,a) with $a : \{\bot,\top\}^n$. Its morphisms $f : \mathrm{El}A\,((m,a),(n,b))$ are morphisms $\mathcal{T}\,(n,m) = \mathcal{T}'(n)^m$ such that $A(f)(b) = (f_1 \bullet b, \ldots, f_m \bullet b) = a$, so

$$\mathrm{El}A\,((m,a),(n,b)) \cong \prod_{1 \le i \le n} \mathrm{El}A\,((1,a_i),(n,b))\,,$$

and

$$(m,a) \cong \coprod_{1 \le i \le n} (1,a_i).$$

In other words, all objects of ElA are generated as coproducts by $(1,\top)$ and $(1,\bot)$. Now, we have $\mathrm{El}A\,((1,a),(n,b)) = \{f : \mathcal{T}'(n) \mid f \bullet b = a.\}$. This is the set of i such that $b_i = a$, and if $a = \top$, it contains $\star$. Note that $\mathcal{T}(n,1) \cong \coprod_{a:A'} \mathrm{El}A\,((n,c),(1,a))$.

Now, for an object $n : \mathcal{T}^{\mathrm{op}}$, the Yoneda embedding sends it to

$$m \mapsto \mathcal{T}\,(n,m) = \mathcal{T}'(n)^m$$

A morphism $f : \mathcal{T}^{\mathrm{op}}\,(n,n') = \mathcal{T}\,(n',n)$ is sent to a natural transformation from $m \mapsto \mathcal{T}\,(n,m)$ to $\mathcal{T}\,(n',m)$ by sending $g : \mathcal{T}\,(n,m) \cong \mathcal{T}'(n)^m$ to

$$g \circ f = (g_1 \bullet f, \ldots, g_n \bullet f) : \mathcal{T}\,(n',m) \cong T'(n)^m.$$

We can calculate (co)limits in a functor category pointwise. That means that

$$(\operatorname*{colim}_{j:J} \mathcal{F}(j))(x) = \operatorname*{colim}_{j:J}(\mathcal{F}(j)(x))$$

We are first and foremost interested in

$$\begin{aligned}\left(\operatorname*{colim}_{(n,a):\mathrm{El}A} Y_{\mathcal{T}}(n)\right)(1) &= \operatorname*{colim}_{(n,a):\mathrm{El}A}(Y_{\mathcal{T}}(n)(1)) \\ &= \operatorname*{colim}_{(n,a):\mathrm{El}A}(\mathcal{T}(n,1)) \\ &= \operatorname*{colim}_{(n,a):\mathrm{El}A}(\mathcal{T}'(n))\end{aligned}$$

which is the equalizer of

$$\coprod_{f:(m,a)\to(m',a')} \mathcal{T}'(m) \rightrightarrows \coprod_{(n,a):\mathrm{El}A} \mathcal{T}'(n)$$

for every $f : (m,a) \to (m',a')$ (i.e. $f : \mathcal{T}'(m')^m$ such that for all i, $f_i \bullet a' = a_i$), we identify the $t : \mathcal{T}'(m)$ over (m,a) with the $t \bullet f : \mathcal{T}'(m')$ over (m',a').

In particular, we identify $\pi_1 : \mathcal{T}'(1)$ over $(1, f \bullet a)$ with $\pi_1 \bullet f = f : \mathcal{T}'(m)$ over (m,a).

For general m and m', we identify $t \bullet f$ over (m',a) with t over $(m, (f_1 \bullet a, \ldots, f_n \bullet a))$. Since the action on the algebra is associative, we have

$$(t \bullet f) \bullet a = t \bullet (f_1 \bullet a, \ldots, f_n \bullet a)$$

so the identifications of the t over (m,a) with π_1 over $(1, t \bullet a)$ generate the equivalence relation.

Now, for the action of $\mathcal{T}$ on this colimit. In particular, given $f : \mathcal{T}(m,1) = \mathcal{T}'(m)$, we want a morphism

$$\mathbf{Set}\left(\left(\underset{(n,a):\mathrm{El}A}{colim}\, Y_{\mathcal{T}}(n)\right)(m), \left(\underset{(n,a):\mathrm{El}A}{colim}\, Y_{\mathcal{T}}(n)\right)(1)\right).$$

We get this as

$$\underset{(n,a):\mathrm{El}A}{colim}\,(t \mapsto f \bullet t) : \mathbf{Set}\left(\underset{(n,a):\mathrm{El}A}{colim}\,(\mathcal{T}'(n)^m),\ \underset{(n,a):\mathrm{El}A}{colim}\,(\mathcal{T}'(n))\right).$$

Every element can be identified with π_1 over $(1,a)$ for some a. Now, for $a_1,\ldots,a_n : A'$, π_1 over $(1,a_i)$ is identified with π_i over $(n,(a_1,\ldots,a_n))$. Then the action of f sends $(\pi_1,\ldots,\pi_n)$ over (n,a) to $f \bullet (\pi_1,\ldots,\pi_n) = f$ over (n,a), but this is identified with π_1 over $(1, f \bullet a)$. Therefore, this recovers the action of $\mathcal{T}'$ on A'.

Week 35

22. The monoid

Note to self: even though everything is defined by actions of Λ*, it might be good to define application and abstraction in* A*.*

In the lambda calculus theory Λ, define

$$\mathbf{1}_n := \lambda x_1 \dots x_{n+1}, x_1 \dots x_{n+1} : \Lambda(0).$$

Specifically, take $\mathbf{1} := \mathbf{1}_1 = \lambda x_1 x_2, x_1 x_2$. Given a Λ-algebra A, we have mappings

$$\begin{aligned} APP &: A \times A \to A & (a_1, a_2) &\mapsto (x_1 x_2) \bullet (a_1, a_2) \\ ABS &: A \to A & a &\mapsto (\lambda x_2, x_1) \bullet a_1 \end{aligned}$$

Also, given any constant $c : \Lambda(0)$, we can inject it as $c \bullet () : A$. We will identify such an element with c. For example, we have the $\mathbf{1}_n : A$. Now, take the sets

$$A(n) = \{a : A \mid APP\ \mathbf{1}_n\ a = a\}.$$

Remark 1. Note that

$$\begin{aligned} APP\ \mathbf{1}_n\ a &= (x_1 x_2) \bullet (\lambda x_1 \dots x_{n+1}, x_1 \dots x_{n+1} \bullet (), a) \\ &= (x_1 x_2) \bullet (\lambda x_2 \dots x_{n+2}, x_2 \dots x_{n+2} \bullet a, x_1 \bullet a) \\ &= (x_1 x_2) \bullet (\lambda x_2 \dots x_{n+2}, x_2 \dots x_{n+2}, x_1) \bullet a \\ &= ((\lambda x_2 \dots x_{n+2}, x_2 \dots x_{n+2}) x_1) \bullet a \\ &= (\lambda x_2 \dots x_{n+1}, x_1 \dots x_{n+1}) \bullet a \end{aligned}$$

which is equal to

$$\begin{aligned} \mathbf{1}_{n-1} \circ a &= (\lambda x_3, x_1 (x_2 x_3)) \bullet ((\lambda x_1 \dots x_n, x_1 \dots x_n) \bullet (), a) \\ &= (\lambda x_3, x_1 (x_2 x_3)) \bullet ((\lambda x_2 \dots x_{n+1}, x_2 \dots x_{n+1}) \bullet a, x_1 \bullet a) \\ &= ((\lambda x_3, x_1 (x_2 x_3)) \bullet ((\lambda x_2 \dots x_{n+1}, x_2 \dots x_{n+1}), x_1)) \bullet a \\ &= (\lambda x_2, (\lambda x_3 \dots x_{n+2}, x_3 \dots x_{n+2})(x_1 x_2)) \bullet a \\ &= (\lambda x_2, (\lambda x_3 \dots x_{n+1}, x_1 x_2 \dots x_{n+1})) \bullet a. \end{aligned}$$

Remark 2. If we have η-equality, $A(1) = A$.

Remark 3. Note that we have $A(n+1) \subseteq A(n)$, because if

$$a = (\lambda x_2 \dots x_{n+2}, x_1 \dots x_{n+2}) \bullet a,$$

then

$$\begin{aligned}(\lambda x_2 \dots x_{n+1}, x_1 \dots x_{n+1}) \bullet a &= (\lambda x_2 \dots x_{n+1}, x_1 \dots x_{n+1}) \bullet ((\lambda x_2 \dots x_{n+2}, x_1 \dots x_{n+2}) \bullet a) \\ &= ((\lambda x_2 \dots x_{n+1}, x_1 \dots x_{n+1}) \bullet (\lambda x_2 \dots x_{n+2}, x_1 \dots x_{n+2})) \bullet a \\ &= (\lambda x_2 \dots x_{n+1}, (\lambda x_2 \dots x_{n+2}, x_1 \dots x_{n+2}) x_2 \dots x_{n+1}) \bullet a \\ &= (\lambda x_2 \dots x_{n+1}, (\lambda x_{n+2}, x_1 \dots x_{n+2})) \bullet a \\ &= (\lambda x_2 \dots x_{n+2}, x_1 \dots x_{n+2}) \bullet a \\ &= a.\end{aligned}$$

We can then define an operation $A(1) \times A(1) \to A(1)$, given by

$$(a_1, a_2) \mapsto a_1 \circ a_2 := (\lambda x_3, x_1(x_2 x_3)) \bullet (a_1, a_2).$$

This works because

$$\begin{aligned}(\lambda x_2, x_1 x_2) \bullet ((\lambda x_3, x_1(x_2 x_3)) \bullet (a_1, a_2)) &= ((\lambda x_2, x_1 x_2) \bullet (\lambda x_3, x_1(x_2 x_3))) \bullet (a_1, a_2) \\ &= (\lambda x_3, (\lambda x_4, x_1(x_2 x_4)) x_3) \bullet (a_1, a_2) \\ &= (\lambda x_3, x_1(x_2 x_3)) \bullet (a_1, a_2).\end{aligned}$$

This associative since

$$\begin{aligned}(a_1 \circ a_2) \circ a_3 &= (\lambda x_3, x_1(x_2 x_3)) \bullet ((\lambda x_3, x_1(x_2 x_3)) \bullet (a_1, a_2), a_3) \\ &= (\lambda x_3, x_1(x_2 x_3)) \bullet ((\lambda x_4, x_1(x_2 x_4)) \bullet (a_1, a_2, a_3), x_3 \bullet (a_1, a_2, a_3)) \\ &= ((\lambda x_3, x_1(x_2 x_3)) \bullet ((\lambda x_4, x_1(x_2 x_4)), x_3)) \bullet (a_1, a_2, a_3) \\ &= (\lambda x_4, (\lambda x_5, x_1(x_2 x_5))(x_3 x_4)) \bullet (a_1, a_2, a_3) \\ &= (\lambda x_4, x_1(x_2(x_3 x_4))) \bullet (a_1, a_2, a_3) \\ &= (\lambda x_4, x_1((\lambda x_5, x_2(x_3 x_5)) x_4)) \bullet (a_1, a_2, a_3) \\ &= ((\lambda x_3, x_1(x_2 x_3)) \bullet (x_1, (\lambda x_4, x_2(x_3 x_4)))) \bullet (a_1, a_2, a_3) \\ &= (\lambda x_3, x_1(x_2 x_3)) \bullet (x_1 \bullet (a_1, a_2, a_3), (\lambda x_4, x_2(x_3 x_4)) \bullet (a_1, a_2, a_3)) \\ &= (\lambda x_3, x_1(x_2 x_3)) \bullet (a_1, (\lambda x_3, x_1(x_2 x_3)) \bullet (a_2, a_3)) \\ &= a_1 \circ (a_2 \circ a_3).\end{aligned}$$

We also have an identity element

$$I = \lambda x_1, x_1$$

since

$$\begin{aligned}a \circ I &= \lambda x_1, a(I x_1) \\ &= \lambda x_1, a x_1 \\ &= a\end{aligned} \qquad \begin{aligned}I \circ a &= \lambda x_1, I(a x_1) \\ &= \lambda x_1, a x_1 \\ &= a.\end{aligned}$$

This makes $A(1)$ into a monoid.

We can also view this as a one-object category M_A. Consider the category $PA := [{M_A}^{\mathrm{op}}, \mathbf{Set}]$ of presheaves on this category. These are sets P, together with a right action $\alpha : P \times M_A \to P$. This category has an object U_A with underlying set $A(1)$ and right action

$$(p, a) \mapsto p \circ a.$$

Now, if we have a map $f : A \to B$ of Λ-algebras, it maps $A(n)$ to $B(n)$, since it preserves the image of $\mathbf{1}_n$ and APP. This gives a map M_f of monoids. By

restriction of scalars, we get a functor $PB \to PA$. It has a left adjoint given by left (or right?) Kan extension.

22.1. The function space.

22.1.1. *For general monoids.* We now study the function space $U_A \Rightarrow U_A$. In general, for any monoid M, given two presheaves $X\ Y : PM$, we have a candidate for the function space $X \Rightarrow Y$. We take it to be the set of M-equivariant maps. I.e. $\varphi : M \times X \to Y$ such that for all $m_1\ m_2 : M$, $x : X$,

$$\varphi(m_1.m_2, x.m_2) = \varphi(m_1, x).m_2.$$

We give this set an M-action:

$$(\varphi.m_3)(m_1, x) = \varphi(m_3.m_1, x)$$

and this is again M-equivariant since

$$(\varphi.m_3)(m_1.m_2, x.m_2) = \varphi(m_3.m_1.m_2, x.m_2) = \varphi(m_3.m_1, x).m_2 = (\varphi.m_3)(m_1, x).m_2$$

This is indeed a right action, since

$$((\varphi.m_3).m_4)(m_1, x) = (\varphi.m_3)(m_4.m_1, x) = \varphi(m_3.m_4.m_1, x) = (\varphi.(m_3.m_4))(m_1, x).$$

The evaluation map $(X \Rightarrow Y) \times X \to Y$ sends (φ, x) to $\varphi(I, x)$. This is universal since if we have a presheaf morphism $e : Z \times X \to Y$, we can create a presheaf morphism $u : Z \to (X \Rightarrow Y)$, given by

$$u(z) = (m, x) \mapsto e(z.m, x)$$

This is an element of $(X \Rightarrow Y)$ because

$$u(z)(m_1.m_2, x.m_2) = e(z.m_1.m_2, x.m_2) = e(z.m_1, x).m2 = u(z)(m_1, x).m_2$$

It is a morphism of presheaves since

$$u(z.m_1)(m_2, x) = e(z.m_1.m_2, x) = u(z)(m_1.m_2, x) = (u(z).m_1)(m_2, x).$$

The map is unique since by the universal property and the fact that $u(z)$ must be a presheaf morphism, we have for all $z : Z$, $m : M$ and $x : X$,

$$u(z)(m, x) = (u(z).m)(I, x) = u(z.m)(I, x) = e(z.m, x).$$

22.1.2. *For our monoid.* We propose $A(2)$ as an alternative candidate for the function space $U_A \Rightarrow U_A$. Since $A(2)$ is characterized by the equation $a = APP\ \mathbf{1}_2\ a$, and that the latter equals $\mathbf{1} \circ a$, so that $A(2)$ has a right action of $A(1)$ (actually a right action of all of A).

We can create a map from $A(2)$ to $U_A \Rightarrow U_A$, which sends an element a_1 to the map $\phi(a_1)$ that sends $(a_2, a_3) \in U_A \times U_A$ to

$$(\lambda x_4, x_1(x_2x_4)(x_3x_4)) \bullet (a_1, a_2, a_3).$$

Note that this is equivariant, since

$$\begin{aligned}
\phi(a_1)(a_2 \circ a_4, a_3 \circ a_4) &= (\lambda x_4, x_1(x_2x_4)(x_3x_4)) \bullet (a_1, a_2 \circ a_4, a_3 \circ a_4) \\
&= (\lambda x_4, x_1(x_2x_4)(x_3x_4)) \bullet (a_1, (\lambda x_3, x_1(x_2x_3)) \bullet (a_2, a_4), (\lambda x_3, x_1(x_2x_3)) \bullet (a_3, a_4)) \\
&= (\lambda x_4, x_1(x_2x_4)(x_3x_4)) \bullet (x_1, (\lambda x_5, x_2(x_4x_5)), (\lambda x_5, x_3(x_4x_5))) \bullet (a_1, a_2, a_3, a_4) \\
&= (\lambda x_5, x_1((\lambda x_6, x_2(x_4x_6))x_5)((\lambda x_6, x_3(x_4x_6))x_5)) \bullet (a_1, a_2, a_3, a_4) \\
&= (\lambda x_5, x_1(x_2(x_4x_5))(x_3(x_4x_5))) \bullet (a_1, a_2, a_3, a_4) \\
&= (\lambda x_5, (\lambda x_6, x_1(x_2x_6)(x_3x_6))(x_4x_5)) \bullet (a_1, a_2, a_3, a_4) \\
&= (\lambda x_3, x_1(x_2x_3)) \bullet ((\lambda x_5, x_1(x_2x_5)(x_3x_5)), x_4) \bullet (a_1, a_2, a_3, a_4) \\
&= (\lambda x_3, x_1(x_2x_3)) \bullet ((\lambda x_4, x_1(x_2x_4)(x_3x_4)) \bullet (a_1, a_2, a_3), a_4) \\
&= ((\lambda x_4, x_1(x_2x_4)(x_3x_4)) \bullet (a_1, a_2, a_3)) \circ a_4 \\
&= \phi(a_1)(a_2, a_3) \circ a_4.
\end{aligned}$$

Also,

$$\phi(a_1 \circ a_4)(a_2, a_3) = \lambda x_1, a_1(a_4(a_2x_1))(a_3x_1) = u(a_1)(a_4 \circ a_2)(a_3),$$

so ϕ is a map of presheaves.

LEMMA 5. *ϕ is an isomorphism.*

PROOF. Take $p = \lambda x_1, x_1(\lambda x_2x_3, x_2)$ and $q = \lambda x_1, x_1(\lambda x_2x_3, x_3)$. These are elements of $A(1)$. Note that for terms c_1, c_2

$$\begin{aligned}
p(\lambda x_1, x_1c_1c_2) &= (\lambda x_1, x_1c_1c_2)(\lambda x_2x_3, x_2) \\
&= (\lambda x_1x_3, x_2)c_1c_2 \\
&= c_1.
\end{aligned}$$

In the same way, $q \circ (\lambda x_1x_2, x_2c_1c_2) = c_2$.

An inverse is given by

$$\psi : f \mapsto \lambda x_1x_2, f(p, q)(\lambda x_3, x_3x_1x_2).$$

This is an inverse, because given $f : U_A \Rightarrow U_A$ and$(a_1, a_2) : U_A \times U_A$, we have

$$\begin{aligned}
\phi(\psi(f))(a_1, a_2) &= u(\lambda x_1x_2, f(p, q)(\lambda x_3, x_3x_1x_2))(a_1, a_2) \\
&= \lambda x_1, (\lambda x_2x_3, f(p, q)(\lambda x_4, x_4x_2x_3))(a_1x_1)(a_2x_1) \\
&= \lambda x_1, f(p, q)(\lambda x_2, x_2(a_1x_1)(a_2x_1)) \\
&= f(p, q) \circ (\lambda x_1, (\lambda x_2, x_2(a_1x_1)(a_2x_1))) \\
&= f(p \circ (\lambda x_1, (\lambda x_2, x_2(a_1x_1)(a_2x_1))), q \circ (\lambda x_1, (\lambda x_2, x_2(a_1x_1)(a_2x_1)))) \\
&= f(\lambda x_1, p(\lambda x_2, x_2(a_1x_1)(a_2x_1)), \lambda x_1, q(\lambda x_2, x_2(a_1x_1)(a_2x_1))) \\
&= f(\lambda x_1, a_1x_1, \lambda x_1, a_2x_1) \\
&= f(a_1, a_2).
\end{aligned}$$

The last line is because $a_i : A(1)$ and therefore $\lambda x_1, a_ix_1 = a_i$.

On the other hand, if we have $a_1 : A(2)$, we have

$$\begin{aligned}\psi(\phi(a_1)) &= \psi((a_2, a_3) \mapsto \lambda x_1, a_1(a_2x_1)(a_3x_1)) \\ &= \lambda x_1x_2, (\lambda x_3, a_1(px_3)(qx_3))(\lambda x_3, x_3x_1x_2) \\ &= \lambda x_1x_2, a_1(p(\lambda x_3, x_3x_1x_2))(q(\lambda x_3, x_3x_1x_2)) \\ &= \lambda x_1x_2, a_1x_1x_2 \\ &= a_1.\end{aligned}$$

The last line is because $a_1 : A(2)$ and therefore $\lambda x_1x_2, a_1x_1x_2 = a_1$.

Therefore, this map is a bijection and an isomorphism. □

From this identification, we get an evaluation map on $A(2)$. First of all, an element $a_1 : A(2)$ is sent to the map

$$(a_3, a_4) \mapsto (\lambda x_1, a_1(a_3x_1)(a_4x_1))$$

which, together with an element $a_2 : A(1)$ is sent to

$$\lambda x_1, a_1(Ix_1)(a_2x_1)$$

for $I = \lambda x_1, x_1$ the identity element of the monoid. This results in an evaluation map

$$A(2) \times A(1) \to A(1) \quad (a_1, a_2) \mapsto \lambda x_1, a_1x_1(a_2x_1).$$

"In the same way", we have $U^n \Rightarrow U \cong A(n+1)$, with evaluation

$$(a, (b_1, \ldots, b_n)) \mapsto \lambda x_1, ax_1(b_1x_1) \ldots (b_nx_1).$$

and since

$$A(n+1) = \{a : A \mid \mathbf{1}_n \circ a = a\},$$

we have a right action of $A(1)$ on this set.

Now, note that for any $a : A(1)$, if we take $a' := \mathbf{1} \circ a$, we have

$$\begin{aligned}\mathbf{1} \circ a' &= \mathbf{1} \circ \mathbf{1} \circ a \\ &= (\lambda x_1x_2, x_1x_2) \circ (\lambda x_1x_2, x_1x_2) \circ a \\ &= \lambda x_1, (\lambda x_2x_3, x_2x_3)(\lambda x_2, x_1x_2) \circ a \\ &= \lambda x_1, (\lambda x_2, (\lambda x_3, x_1x_3)x_2) \circ a \\ &= (\lambda x_1x_2, x_1x_2) \circ a \\ &= \mathbf{1} \circ a \\ &= a',\end{aligned}$$

so we have a mapping $A(1) \to A(2)$, and an injection $A(2) \hookrightarrow A(1)$. Since for all $a : A(2)$, $\mathbf{1} \circ a = a$, we have that composition on the left with $\mathbf{1}$ gives a retraction from $A(1)$ onto $A(2)$ and the endomorphism theory of $A(1)$ in PA is a λ-theory. We call this $\mathcal{U}_A$.

CHAPTER 2

Week 36

1. The Karoubi envelope

Given a category $\mathcal{C}$, define a category $\overline{\mathcal{C}}$, with objects being pairs (A, e), with $e : A \to A$ an idempotent. A map $(A, e), (B, f)$ is a map $v : A \to B$ such that $f \circ v \circ e = v$. The identity on (A, e) is e, since if $f \circ v \circ e = v$, then $v \circ e = f \circ v \circ e \circ e = f \circ v \circ e = v$ (and likewise for $f \circ v = v$).

We can embed $\mathcal{C}$ into $\overline{\mathcal{C}}$ by sending $A : \mathcal{C}_0$ to (A, Id_A), and $f : \mathcal{C}(A, B)$ to f, since $Id_B \circ f \circ Id_A = f$ is trivial.

Given an idempotent $e : \mathcal{C}(A, A)$, take $r = e : \overline{\mathcal{C}}((A, Id_A), (A, e))$ and $s = e : \overline{\mathcal{C}}((A, e), (A, Id_A))$. Then $r \circ s = e \circ e = e = Id_{(A,e)}$ and $s \circ r = e \circ e = e$. Therefore, e splits in $\overline{\mathcal{C}}$. This also shows that any object $(A, e) : \overline{\mathcal{C}}_0$ is a retract of (the image of) an object in $\mathcal{C}$. That is why the paper refers to $\overline{\mathcal{C}}$ as "the category of retracts".

2. Relatively cartesian closed

Hyland gives a new proof that $\overline{(\mathcal{L}(1))}$ is cartesian closed (when viewing $\mathcal{L}(1)$ as a monoid with operation $\bullet$, and then as a one-object category), based on the proof in [**Tay86**].

For a Λ-algebra A, we can take $A(1)$, the set of functional elements of A. This forms a monoid under composition: $a; b = Pab$ for $P = \lambda fgx, g(fx)$. The Karoubi completion of this category has as objects the terms a such that $Paa = a$. The morphisms $a \to b$ are elements $f : A$ such that $Paf = f = Pfb$.

Since we view $\overline{A(1)}$ as a category, we can try to define products and exponentials of elements $A\ B : \overline{A(1)}$. This is done in §§1.3.5 and 1.3.6.

Taylor defines a family $(X_a)_{a:A}$ to be the *display map* $f : X \to A$ (the X_a arise as preimages/pullbacks of the $a : A$). We have a category $\mathcal{C}^A$, consisting of families over A. This is the slice category $\mathcal{C}/A$. If we have a morphism $f : A \to B$, we get a functor $\mathcal{C}^f : \mathcal{C}^B \to \mathcal{C}^A$, which is the pullback along f, and we will call it f^* or $\mathtt{P}f$. For a diagram

$$\begin{array}{ccc} Y & & X \\ \downarrow & & \downarrow \\ A & \xrightarrow{f} & B \end{array}$$

there is a natural bijection between morphisms $Y \xrightarrow{f} X$ over f, and morphisms $Y \to f^*X$.

Pullbacks give binary products in $\mathcal{C}^A$. That is why these are also called "fibred products".

Now, for indexed products: In **Set**, given indexed sets $(X_a)_{a:A}$, we have the product $\prod_{a:A} X_a$. Its elements are indexed families of elements $(x_a)_{a:A}$. If we have

an element $X : \mathcal{C}^A$, the elements of $\prod_{a:A} X_a$ correspond to maps $\mathcal{C}^a\,(A, X)$, i.e., maps that send a to an element in X_a.

The indexed product with a constant set $\prod_{a:A} B$ is the exponential B^A. Therefore, if a category has indexed products over itself, it is cartesian closed.

We can also do this in an indexed fasion. If we have a family B indexed over A (given by a morphism $f : A \to B$), and for every $a : A$, a family $(X_b)_{b:B_a}$ (equivalent to a morphism $X \to B$), then we can construct the products $(\prod \alpha X)_a := (\prod_{b:B_a} X_b)$ for all $a : A$. That means that we have the type $\prod \alpha X$ over A.

A morphism in $\mathcal{C}^A$ from C to $(\prod \alpha X)$ is equivalent to a collection of morphisms $\varphi_a : C_a \to (\prod \alpha X)_a = (\prod_{b:B_a} X_b)$. This is equivalent to a collection of morphisms $\varphi_{ab} : C_a \to X_b$ for all $a : A$ and $b : B_a$. Note that $a = \alpha b$, so $C_a = C_{\alpha b} = (\alpha^* C)_b$. So for all $b : B$, we need a morphism $(\alpha^* C)_b \to X_b$. However, this is a morphism $\alpha^* C \to X$ over B. Therefore,

$$\mathcal{C}^A \left(C, \prod \alpha X\right) \cong \mathcal{C}^B \left(\alpha^* C, X\right)$$

and the functor $\prod \alpha : \mathcal{C}^B \to \mathcal{C}^A$ is a right adjoint to the functor $\alpha^* : \mathcal{C}^A \to \mathcal{C}^B$.

For a morphism $B \to A$ and a morphism $X \to B$, we can take the indexed sums $(\sum \alpha X)_a := (\sum_{b:B_a} X_b)_a$.

A morphism $\sum \alpha X \to C$ in $\mathcal{C}^A$ is equivalent to a family of morphisms $\varphi_a : \sum_{b:B_a} X_b \to C_a$. This is equivalent to, for every $a : A$ and every $b : B_a$, a morphism $\varphi_{ab} : X_b \to C_a = (\alpha^* C)_b$. This is a morphism $X \to \alpha^* C$ in $\mathcal{C}^B$, so $\sum \alpha \dashv \alpha^*$.

Therefore, Taylor defines an internal product in an indexed category to be a right adjoint to substitution over a display map, and an internal sum to be a left adjoint.

...

Hyland's view of this matter is as follows:

Given a λ-theory $\mathcal{L}$, the category $\mathbb{R}$ is the category of retracts of the idempotent elements of $\mathcal{L}(0)$ (I think). That is: the objects of $\mathbb{R}$ are elements $a : \mathcal{L}(0)$, such that $\lambda x, a(ax) = a$. Morphisms between idempotents a and b are elements $f : \mathcal{L}(0)$ such that $\lambda x, (b(f(ax))) = f$.

3. The Curry Festschrift

The Lambda calculus is a theory of functions. It specifies what one can do with functions (in a (concrete) cartesian closed category): One can abstract a function $A \times B \to C$ to get a function $A \to (B \to C)$; One can apply a function $A \to B$ to an element of A to get an element of B; And one always has the identity function $x : A \to A$.

For the typed lambda calculus, the objects in $\mathcal{C}$ form the types, and the morphisms (elements of the exponential object) are the lambda terms.

For the untyped lambda calculus, we can say that all terms have the same domain and codomain $U \to V$. However, since we can compose them with themselves, we must have $U = V$. Also, given two terms $f\ g : U \to U$, we can apply f to g, so $(U \to U) = U^U$ must be inside U. Therefore, we want a section $U^U \to U$ and a retraction $U \to U^U$.

4. Comparing Hyland, Taylor and Curry

4.1. Taylor. Taylor defines a *Combinatory prealgebra* to be a set Λ with a binary operation $\bullet$ ("application") and elements K $(\lambda xy, x)$ and S $(\lambda xyz, xz(yz))$ such that for all $a\ b\ c : \Lambda$, $Kab := K \bullet a \bullet b = a$ and $Sabc = ac(bc)$.

For $f : \Lambda$, he defines f to be *functional* if it has a representation $f = \lambda x, a$.

He defines a *Combinatory algebra* to be a prealgebra Λ such that if $fx = gx$ (in $\Lambda[x]$, i.e., Λ to which we adjoin an additional element x), then $f = g$.

He defines a *model* to be a combinatory algebra Λ, such that for all $f\ g : \Lambda$, if for all $a : \Lambda$, $fa = ga$, then $f = g$.

He then defines $\mathbf{Retr}(\Lambda)$ to be the category of retracts of the monoid of functional elements in Λ. An object is an element $A : \Lambda$ such that $PAA = A$, and a morphism $A \to B$ is an element $\alpha : \Lambda$ such that $PA\alpha = \alpha = P\alpha B$.

He notes that we can view objects (idempotents) of $\mathbf{Retr}(\Lambda)$ as types $\|A\| = \{a \mid a = Aa\} \subseteq \Lambda$ and morphisms $\alpha : \mathbf{Retr}(\Lambda)(A, B)$ as maps $a \mapsto \alpha a : \|A\| \to \|B\|$.

At the end of chapter 1, he shows that $\mathbf{Retr}(\Lambda)$ is cartesian closed.

In chapter 4, Taylor introduces a fibration. If $\mathcal{C}$ is a category and D is a class of morphisms, we say that D is a class of *Display Maps* for $\mathcal{C}$ if D is closed under pullbacks along $\mathcal{C}$-morphisms, under composition and the maps to the terminal object are in D.

Then, given any $A : \mathcal{C}$, we can construct $\mathcal{C}/_{\mathcal{D}}A$ as a full subcategory of $\mathcal{C}/A$, containing as objects only the display maps to A. The $\mathcal{C}/_{\mathcal{D}}A$ form a fibration over $\mathcal{C}$.

For $\alpha : \mathcal{C}(A, B)$, we have $\alpha_* : \mathcal{C}/_{\mathcal{D}}A \to \mathcal{C}/_{\mathcal{D}}B$ ("the substitution functor"). It has a left adjoint, given by postcomposition, which we will call $\sum \alpha$. If it also has a right adjoint, (which we then call $\prod \alpha$), we call the category relatively cartesian closed.

Taylor notes that $\mathcal{C}/_{\mathcal{D}}1$, the fiber over the terminal object, consists of all of $\mathcal{C}$. He also notes that a category that is relatively cartesian closed has cartesian closed fibers.

4.2. Hyland. Hyland defines an algebraic theory to be a functor $\mathcal{T} : F \to \mathbf{Set}$, together with variables $\pi_i : \mathcal{T}(n)$ for all $1 \leq i \leq n$, and an operation ("substitution") $\bullet : \mathcal{T}(m) \times \mathcal{T}(n)^m \to \mathcal{T}(n)$ which is associative, unital, dinatural.

He defines a λ-theory to be an algebraic theory $\mathcal{L}$, together with retractions $\rho : \mathcal{T}(n) \to \mathcal{T}(n+1)$, that are compatible with $\bullet$ (which, on $\mathcal{T}(n+1)$, leaves alone the $n+1$th component).

He defines Λ to be the λ-theory corresponding to the λ-calculus, with $\Lambda(n)$ the set of lambda terms in n indeterminates.

Finally, he defines a Λ-algebra (one would call this a "model") to be a set A, together with an action $*(n) \times A^n \to A$ for all n that is associative and unital.

4.3. Curry. Curry does not give a formal characterization of the λ-calculus (he refers to 'Lambek's paper'). However, he notes that there is a correspondence between cartesian closed categories, and 'extensional typed λ-calculi'.

This means that a typed λ-calculus has at least an identity function $x_1 : A \to A$. He also notes that, given a term $f : A \times B \to C$, we can abstract it to $\lambda b, f : A \to (B \to C)$. Finally, given variables $u : A \to B$ and $v : A$, we have $u(v) : (A \to B) \times A \to B$. Therefore, we have abstraction and application.

He also notes that we need extensionality: if $f = g$, then $\lambda x, f = \lambda x, g$.

For the untyped λ-calculus, he constructs the terms 'in the usual way', by starting with variables $x_1, \ldots$ and then using abstraction and application.

He then constructs the category $\mathbb{R}$ straightforwardly, with as objects the terms A without free variables, such that $A = \lambda x, A(Ax)$. The maps $f : A \to B$ are terms f without free variables, for which we can prove $f = \lambda x, B(f(Ax))$.

He also constructs products $A \times B := \lambda uz, z(A(u(\lambda xy, x)))(B(u(\lambda xy, y)))$, with projections

$$p_{AB} := \lambda u, (A \times B)(u)(\lambda xy, x) \quad \text{and} \quad q_{AB} := \lambda u, (A \times B)u(\lambda xy, y),$$

with the product of morphisms $f : C \to A$ and $g : C \to B$ being $\langle f, g \rangle := \lambda tz, z(ft)(gt)$.

Finally, he defines the function space to be

$$A \to B := \lambda f, B \circ f \circ A$$

with evaluation

$$\varepsilon_{BC} = \lambda u, C(u(\lambda xy, x))(B(u(\lambda xy, y)))$$

and for $h : A \times B \to C$,

$$\Lambda_{ABC} h = \lambda xy, h(\lambda z, zxy).$$

He takes the reflexive object $U := \lambda x, x$.

Bibliography

[ARV11] J. Adámek, J. Rosický, and E. M. Vitale. *Algebraic theories*, volume 184 of *Cambridge Tracts in Mathematics*. Cambridge University Press, Cambridge, 2011. A categorical introduction to general algebra, With a foreword by F. W. Lawvere.

[FTBF23] Zachary Flores, Angelo Taranto, Eric Bond, and Yakir Forman. A formalization of operads in coq, 2023.

[Lei03] Tom Leinster. Higher operads, higher categories, 2003.

[Tay86] Paul Taylor. *Recursive Domains, Indexed Category Theory and Polymorphism*. PhD thesis, University of Cambridge, 1986.