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
```

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].`.

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?

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

CHAPTER 2

Week 08

1. 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.

2. 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**]).

3. Operads

3.1. 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?).

3.2. 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 $f \circ (f_1, \ldots, f_m) \in X'(n_1 + \cdots + n_m)$.

4. T-operads

4.1. 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.

4.2. 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.

5. Cartesian Operads

5.1. 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 $F \odot G = \int^{n \in \mathbf{Fin}} F(n) G^n$.

5.2. 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.

5.3. 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.

CHAPTER 3

Week 09

1. Adjunctions

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

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

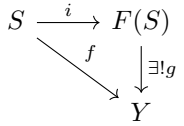
that is natural in X and Z. This is an example of a general notion:

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 $\mathcal{C}(FX, Y) \cong \mathcal{D}(X, GY)$ that is dinatural in X and Y.

1.1. 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 $\mathcal{C}(F(X), Y) \cong \mathbf{Set}(X, G(Y))$, 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: $f \in \mathbf{Set}(S, GY)$). Then the fact that $\mathbf{Set}(FS, Y) \cong \mathbf{Set}(S, GY)$ 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 $S-\mathbf{Mod} \to R-\mathbf{Mod}$. This has a left adjoint, $M \mapsto M \otimes_R S$.

2. Operads and cartesian operads

2.1. 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 $f \in P(m)$, $g_1 \in P(n_1)$, …, $g_m \in P(n_m)$,

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

2.2. 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

3. 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

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

4. 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.

CHAPTER 4

Week 11

1. 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.

CHAPTER 5

Week 17

1. 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 bijection $\varphi : \mathbf{Set}(\{1, \dots, n\}, F(c)) \xrightarrow{\sim} \mathcal{C}(G(\{1, \dots, n\}), c)$. 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 $f \in \mathcal{C}(A, B)$, we have a function $F(f) \in \mathbf{Set}(F(A), F(B))$ (**TODO: Show that it is an algebra morphism**). 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 $T(n) = \{1, \dots, n\}$. Take $(A, B) \in \mathbf{Set} \times \mathbf{Set}_0$. Then $F(A, B) = A$. Also, for all n and all $a : \{1, \dots, n\} \to A$, $\varphi(a) = (a, \iota_\emptyset)$ 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.

2. 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.

Bibliography

[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.