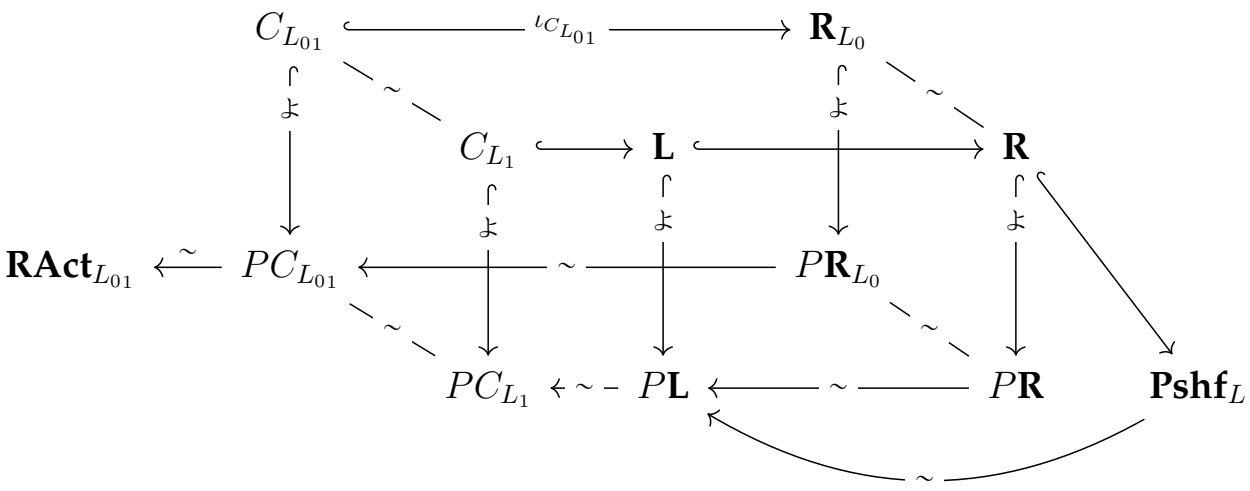
Universal algebra, Univalent foundations and the Untyped λ-calculus

Version of October 11, 2024



Arnoud van der Leer

Universal algebra, Univalent foundations and the Untyped λ-calculus

THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Arnoud van der Leer
born in Leiderdorp, the Netherlands

Programming Languages Group
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
www.ewi.tudelft.nl

Cover picture: A 2-commutative diagram featuring categories that appear in this thesis.

Universal algebra, Univalent foundations and the Untyped λ-calculus

Author: Arnoud van der Leer
Student id: 4485890
Email: a.a.vanderleer@student.tudelft.nl

Abstract

Abstract here.

Thesis Committee:

Chair:	Prof. dr. C. Hair, Faculty EEMCS, TU Delft
Committee Member:	Dr. A. Bee, Faculty EEMCS, TU Delft
Committee Member:	Dr. C. Dee, Faculty EEMCS, TU Delft
University Supervisor:	Ir. E. Ef, Faculty EEMCS, TU Delft

Contents

Chapter 1

Introduction

The λ-calculus is an abstract tool that is used in an area where mathematics meets computer science, to study algorithms, programming languages and even category theory. It was conceived by Alonzo Church, primarily as a foundation for mathematics instead of set theory or type theory [Chu32]. A couple of years later, Church used it to show that the 'Entscheidungsproblem' was unsolvable: the problem asked for an algorithm that could tell about any mathematical statement whether it was true or false, and Church showed that such an algorithm could not exist [Chu36]. A year later, Alan Turing showed, using previous work of Kleene [Kle36], that an algorithm is definable using the λ-calculus if and only if it is definable using a Turing machine [Tur37], solidifying the position of both the λ-calculus and Turing machines as ways to talk about algorithms.

Later, all kinds of different flavours and extensions of the λ-calculus were put forth, with colorful names like 'simply typed λ-calculus', 'System T' and 'PCF'. Even though the λ-calculus was originally a very theoretical tool, it was also the inspiration for function programming languages, and traces of it can be seen in imperative programming languages, where unnamed functions are commonly called 'lambda expressions' [Ora22] and are sometimes even written like `lambda x y : (x - y) * (x + y)` [Fou24].

Even so, the theoretical study of the λ-calculus and its extensions continues to this day. For example, in 2017[1], a paper by Martin Hyland was published with the title 'classical lambda calculus in modern dress' [Hyl17]. In this paper, Hyland approaches the λ-calculus from the viewpoint of universal algebra, using algebraic theories, and more generally category theory, to study it. This way, he obtains two new proofs for old theorems. The paper also contains a new theorem that shows that two different ways to study the λ-calculus using universal algebra are equivalent.

Now, in the last century, mathematics has changed a lot. Of course, new theorems have been proved, new conjectures have been made and entire new areas of mathematics have come into existence. However, as in many professions, the arrival of the computer has affected the way that work is done in mathematics. All kinds of tools have been created that aid mathematicians in their job. Some of these tools help with quick calculations, for formulating or disproving new conjectures. Other tools help in structurally verifying ideas. There even have been some 'proofs by computer', which consist of a proof on paper that a theorem can be reduced to a finite, but very large, computation, and a computer program that then executes this computation. For example, the first proofs of the four color theorem [AH76] and the Kepler conjecture [Hal02] were done this way.

However, because the computation part of such a proof involves a lot of code, and computer code tends to contain bugs, accepting a proof by computer involves a certain amount of trust. Therefore, both of the theorems mentioned have subsequently also been proved using 'computer proof assistants'[Gon08][HAL+17]. Such proof assistants, like 'rocq', 'lean' and

[1] Note that the paper has been around since 2012, when it was first published as a preprint on arXiv.

'Agda', are computer programs with only a very small 'trusted codebase'[2], that can verify mathematical reasoning. In this way, if we trust the small core of such a proof assistant, and the proof assistant says that a proof is correct, then we can trust that indeed, the proof is correct.

This sounds great in theory, but in practice, the way that these programs reason about mathematics is very formal and rigid. This means that 'formalizing', proving something using a proof assistant, usually involves a great amount of effort, usually much more than doing a pen-and-paper proof. However, representing this as a choice between doing pen-and-paper proofs and working with proof assistants would be too simplistic. Often, doing a pen-and-paper proof can help one to develop an intuition and get new ideas, whereas formalizing those ideas can then help to get a better view of the subtleties of a proof and to sharpen the understanding of why the proof works. Therefore, there is a lot of ongoing research into tools and best practices to make the process of formalizing as smooth as possible; to make these programs, which sometimes are experienced as 'proof obstructors', really into 'proof assistants'.

Now, much of contemporary mathematics is built on the foundation of set theory, which we will often refer to as *classical mathematics*. Usually, 'set theory' refers to ZFC: Zermelo Fraenkel set theory with the axiom of choice, developed in the early 1900s. On the other hand, most of the formalization in proof assistants is built on the foundation of type theory. This is because computers can reason better with types than with sets, as evidenced by the many typed programming languages that are around nowadays. However, type theory has been around longer than computers have: it was initially developed by Bertrand Russell in the early 1900s, to create a foundation of mathematics that avoided Russell's paradox[3].

Nowadays, there are many flavours of type theory around. The proof assistant rocq is based on the 'calculus of constructions' whereas the proof assistant Agda works with the 'unified theory of dependent types'. Every flavour has its advantages and disadvantages, but in this thesis we will work with 'univalent foundations', set forth by Vladimir Voevodsky [Voe14]. In univalent foundations, the common mathematical principle that 'things with a similar structure[4], are the same' is made explicit. The principle is used often in modern mathematics, and making this explicit in the type theory that we work with, allows us to use the principle when formalizing.

However, note that two objects can often have a similar structure in more than one way. For example, the set $\{\top, \bot\}$ has two bijections to the set $\{0, 1\}$. To accomodate for this, in univalent foundations, two things can sometimes 'be the same' in more way than one. For example, we have two equalities between $\{\top, \bot\}$ and $\{0, 1\}$. This means that the concept of 'sameness' is more intricate to work with in univalent foundations than it is in set theory. This is one of the reasons that there are subtleties involved in transferring definitions and proofs from set theory to univalent foundations. It is therefore still a topic of ongoing research how well different parts of set-based mathematics can be transferred to univalent foundations and what subtleties pop up when material from set-based mathematics is transferred to univalent foundations. For example, often in set-based mathematics, two definitions are equivalent if we assume the axiom of choice. However, sometimes in univalent foundations, the axiom of choice is not sufficient to make two such definitions equivalent, and then we have two different definitions. It is then interesting to explore the behaviour of these two different definitions in univalent foundations.

[2]Trusted codebase means the part of the codebase of which we hope that it is correct. The rest of the codebase is checked using this trusted core, so if the trusted core is correct, the rest is too.

[3]"Consider the set of all sets that do not contain themself. Does this set contain itself?"

[4]For example, two sets with a bijection between them have a similar structure. Also, two graphs G and H have a similar structure when there is a bijection f between their sets of vertices, such that for every two vertices $v, w : G$, we have a bijection between the set of edges from v to w, and the set of edges from $f(v)$ to $f(w)$.

In this thesis, we study the paper 'classical lambda calculus in modern dress' through the lens of univalent foundations, work out the details of the proofs and formalize part of the paper and its preliminaries. The major contributions are

- The category theoretical preliminaries needed to understand the paper (Chapter 2), as well as the work of Hyland's predecessors that he expands upon (Chapter 5), and more detailed versions of his definitions and proofs, complemented with some examples (Chapters 4 and 6). This makes Hyland's paper more accessible to computer scientists.
- The translation of Hyland's paper and the work of his predecessors to univalent foundations (Chapter 6). This shows that his work can be translated to univalent foundations, and contributes to the knowledge about translating classical mathematics to univalent foundations in general.
- A formalization of part of the paper in rocq (Chapter 7). This contributes to the knowledge about formalization in general.
- A new proof for the fundamental theorem of the λ-calculus (Section 6.4). This version is easier to read and verify, because it is more elementary and uses less category theory than Hyland's proof.
- An analysis of the behaviour of the Karoubi envelope in univalent foundations in general (Section 2.12), and in the specific case of Paul Taylor's and Martin Hyland's work (Remark 6.11). Here we see an example of two definitions that are equivalent in classical mathematics, but become different in univalent foundations. It is interesting to see the differences between the behaviour of the two definitions in univalent foundations.

Now, most of this thesis works towards Chapter 6, about the three main proofs of Hyland's paper. Because this thesis works from a univalent point of view, Chapter 3 introduces univalent foundations and builds the preliminary knowledge in that area for the rest of the paper. As mentioned before, the work of Hyland's predecessors that is covered in Chapter 5 helps to understand his paper. The category theoretical preliminaries for understanding Hyland's paper are covered in Chapter 2. It is probably wise to skim this chapter, and periodically come back to it to better understand the material in the other chapters. Hyland's three main theorems deal with a couple of objects that he introduces. These objects, together with some properties and examples, are introduced in Chapter 4. Lastly, the formalization of part of the material is discussed in Chapter 7.

Throughout this document, links are included to `documentation` of the corresponding formalized material. The documentation refers to the state of `the UniMath repository` at commit `5eb5c8958c4dddd4219f895bf7bc51547395522d`.

Chapter 2

Category Theoretic Preliminaries

I will assume a familiarity with the category-theoretical concepts presented in [AW23]. These include categories, functors, isomorphisms, natural transformations, adjunctions, equivalences and limits.

2.1 Notation

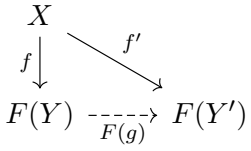
Throughout this thesis, objects of a homotopy set, for example morphisms, will usually be denoted with lowercase letters like f, g, h and sometimes with Greek letters like $\epsilon, \eta, \varphi, \psi$. Objects of a homotopy 1-type, like objects in a category, will usually be denoted with capital letters like X, Y, Z. Lastly, (displayed) (bi)categories themselves will usually be denoted with boldface capital letters like **Set** or **D**.

Throughout this thesis, an object X in a category $\mathbf{C}$ will be denoted by $X : \mathbf{C}$. A morphism f between objects X and Y in a category $\mathbf{C}$ will be denoted by $f : \mathbf{C}(X, Y)$ or sometimes $f : X \to Y$. Composition of morphisms $f : \mathbf{C}(X, Y)$ and $g : \mathbf{C}(Y, Z)$ will be denoted by $f \cdot g$. And composition of functors $F : \mathbf{A} \to \mathbf{B}$ and $G : \mathbf{B} \to \mathbf{C}$ will be denoted by $F \bullet G$.

2.2 Universal Arrows

One concept in category theory that can be used to describe a lot of limits and adjunctions is that of a universal arrow (see for example [Mac98], Part III)

Definition 2.1. A *universal arrow* from an object $X : \mathbf{D}$ to a functor $F : \mathbf{C} \to \mathbf{D}$ consists of an object $Y : \mathbf{C}$ and a morphism $f : \mathbf{D}(X, F(Y))$ such that for every similar pair (Y', f'), f' factors uniquely as $f \cdot F(g)$ for some $g : \mathbf{C}(Y, Y')$:



Alternatively, we can characterize universal arrows by their action on hom-sets:

Lemma 2.2. *Let $F : \mathbf{C} \to \mathbf{D}$ be a functor and $X : \mathbf{D}$ an object. An object $Y : \mathbf{C}$ and an arrow $f : \mathbf{D}(X, F(Y))$ form a universal arrow from X to F if and only if the function*

$$(g \mapsto f \cdot F(g)) : \mathbf{C}(Y, x) \cong \mathbf{D}(X, F(x))$$

is a bijection.

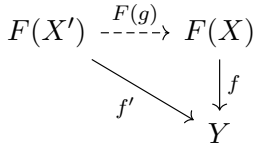
Conversely, for all $Y : \mathbf{C}$ and $X : \mathbf{D}$, every bijection

$$\mathbf{C}(Y, Z) \cong \mathbf{D}(X, F(Z))$$

that is natural in Z arises in this way from some universal arrow $f : \mathbf{D}(X, F(Y))$.

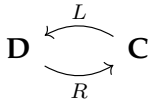
Proof. See [Mac98], Chapter III.2, Proposition 1. □

There is also the dual concept: a universal arrow (X, f) from a functor F to an object $Y : \mathbf{C}$. Its universal property can be summarized in the following diagram:



2.3 Adjunctions and Equivalences

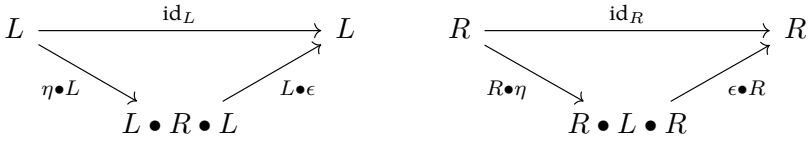
Recall that an *adjunction $L \dashv R$* is a pair of functors



with natural transformations (the unit and co-unit)

$$\eta : \mathrm{id}_{\mathbf{C}} \Rightarrow L \bullet R \quad \text{and} \quad \epsilon : R \bullet L \Rightarrow \mathrm{id}_{\mathbf{D}}$$

such that the diagrams



commute (these are called the *triangle identities* or *zigzag identities*). Here the natural transformation $\eta \bullet L : L \bullet R \bullet L$ is the natural transformation η whiskered on the right by L, and the other whiskered transformations are similar.

An alternative characterization (see [Mac98], chapter IV.1, Theorem 2) of an adjunction $L \dashv R$ is as a natural bijection

$$\varphi : \mathbf{D}(L(X), Y) \xrightarrow{\sim} \mathbf{C}(X, R(Y)).$$

Naturality means that for all $f : \mathbf{C}(X', X)$, $g : \mathbf{D}(Y, Y')$ and $h : \mathbf{D}(L(X), Y)$,

$$\varphi(L(f) \cdot h \cdot g) = f \cdot \varphi(h) \cdot R(g).$$

Lastly, one can construct an adjunction using universal arrows. This lends itself particularly well for a formalization, where it is often preferable to have as few 'demonstranda' as possible:

Lemma 2.3. *One can construct an adjunction (L, R, η, ϵ) as above from only the functor $L : \mathbf{C} \to \mathbf{D}$ and, for each $X : \mathbf{C}$, a universal arrow $(R(X), \epsilon_X)$ from L to X.*

Proof. See [Mac98], Chapter IV.1, Theorem 2 (iv). □

2.3.1 Adjoint equivalences

An (adjoint) equivalence of categories has two equivalent definitions, that only differ in which notion they take as the base. In this case, we will use the definition that is based on adjunctions:

Definition 2.4. An *adjoint equivalence* between categories $\mathbf{C}$ and $\mathbf{D}$ is a pair of adjoint functors $L \dashv R$ like above such that the unit $\eta : \mathrm{id}_{\mathbf{C}} \Rightarrow L \bullet R$ and co-unit $\epsilon : R \bullet L \Rightarrow \mathrm{id}_{\mathbf{D}}$ are isomorphisms of functors.

The alternative is to define an adjoint equivalence as an equivalence of categories (a tuple (L, R, η, ϵ) of two functors and two natural transformations, where η and ϵ are isomorphisms of functors) that additionally satisfies the zigzag identities.

2.3.2 Weak equivalences

There is also the notion of 'weak equivalence'. In some cases, this is equivalent to an adjoint equivalence (for example, when its domain is univalent, see Section 3.2).

Definition 2.5. A functor $F : \mathbf{C} \to \mathbf{D}$ is called a *weak equivalence* if it is essentially surjective and fully faithful.

2.3.3 Exponential objects

Note that in the category of sets, for all $X, Y : \mathbf{Set}$, we have a set of functions $(X \to Y)$. Also, for all X, Y, Z, there is a (natural) bijection

$$(X \times Y \to Z) \cong (X \to (Y \to Z))$$

which we can also write as

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

In other words, we have functors $X \mapsto X \times Y$ and $Z \mapsto (Y \to Z)$, and these two form an adjunction. The following generalizes this

Definition 2.6. A category $\mathbf{C}$ has *exponential objects* (or *exponentials*) if for all $X : \mathbf{C}$, the functor $X' \mapsto X' \times X$ has a right adjoint, which we denote $Y \mapsto Y^X$.

Remark 2.7. It is actually very well possible that a category does not have all exponentials, but it has some objects $X, Y, Y^X : \mathbf{C}$ with a bijection

$$\mathbf{C}(X' \times X, Y) \cong \mathbf{C}(X', Y^X)$$

that is natural in X'. Then Y^X is still called an exponential object.

2.3.4 Forgetful functors and free objects

In mathematics, we often deal with objects that are 'based on' other objects. For example, a ring is a set with some additional structure. Often, this is a relation between the respective categories (for example, in the case of a displayed category, see Section 7.4), and such a relation gives rise to a *forgetful functor*, that 'forgets' about the additional structure. In the examples of rings and sets, the forgetful functor sends a ring to its underlying set, and a ring morphism to the function between the sets. However, note that there is no formal definition of forgetful functors. The name is more of a way to talk about the perceived relation between the categories.

Definition 2.8. Given a forgetful functor $F : \mathbf{C} \to \mathbf{D}$, we define the *free functor* associated to F to be the left adjoint to F, if it exists.

Example 2.9. Consider the forgetful functor from the category of commutative rings to the category of sets, sending a ring to its underlying set. This has a left adjoint, sending the set $\{1, 2, \ldots, n\}$ to the polynomial ring $\mathbb{Z}[X_1, \ldots, X_n]$, and more generally, sending S to the polynomial ring $\mathbb{Z}[X_s]_{s:S}$. This ring is then called 'the free commutative ring on S'. If S has n elements, the ring is also called 'the free commutative ring on n generators'.

The free functor sends a function $f : S \to T$ to the ring morphism $\mathbb{Z}[X_s]_{s:S} \to \mathbb{Z}[X_t]_{t:T}$ that sends X_s to $X_{f(s)}$.

The natural bijection

$$\mathbf{Rng}(\mathbb{Z}[X_s]_{s:S}, R) \cong \mathbf{Set}(S, R)$$

then sends $f : \mathbf{Rng}(\mathbb{Z}[X_s]_{s:S}, R)$ to $s \mapsto f(X_s)$ and $g : \mathbf{Set}(S, R)$ to the morphism that sends X_s to $g(s)$.

However, as with exponential object, sometimes we have a forgetful functor $F : \mathbf{C} \to \mathbf{D}$, but we cannot give a free functor on the entire category $\mathbf{D}$. In such a case, we might still talk about free 'objects':

Definition 2.10. Let $F : \mathbf{C} \to \mathbf{D}$ be a forgetful functor. Given $X : \mathbf{D}$, the *free object* on X is a universal arrow (Y, f) from X to F.

Remark 2.11. By [Mac98], Chapter IV.1, Theorem 2 (ii), if we have a free object on every $X : \mathbf{D}$, we can piece these together to get a free functor associated to F.

2.4 Yoneda

Let $\mathbf{C}$ be a category. One of the categories that is often important in the study of $\mathbf{C}$ is its *presheaf category* $P\mathbf{C} = [\mathbf{C}^{\mathrm{op}}, \mathbf{Set}]$. Its objects are the functors from C^{op} to $\mathbf{Set}$ and its morphisms are the natural transformations between the functors.

We can embed $\mathbf{C}$ fully faithfully into $P\mathbf{C}$ as follows (see [KS06], Section 1.4):

Definition 2.12. The *Yoneda embedding* $よ : \mathbf{C} \hookrightarrow P\mathbf{C}$ is given on objects by $よ(Y) = \mathbf{C}(-, Y)$:

$$よ(Y)(X) = \mathbf{C}(X, Y) \quad \text{and} \quad よ(Y)(f)(g) = f \cdot g$$

for $X : \mathbf{C}$, $f : \mathbf{C}(X, X')$ and $g : \mathbf{C}(X', Y)$. It sends a morphism $f : \mathbf{C}(Y, Y')$ to the natural transformation $よ(f) : \mathbf{C}(-, Y) \Rightarrow \mathbf{C}(-, Y')$ given by

$$よ(f)(X)(g) = g \cdot f$$

for $X : \mathbf{C}$ and $g : \mathbf{C}(X, Y)$.

Now, this embedding has a couple of properties:

Lemma 2.13. *For any $Y : \mathbf{C}$ and $F : P\mathbf{C}$, we have an equivalence $P\mathbf{C}(よ(Y), F) \simeq F(Y)$, and this equivalence is natural in Y and F.*

Proof. It sends a natural transformation $a : P\mathbf{C}(よ(Y), F)$ to the element $a_Y(\mathrm{id}_Y)$. Conversely, it sends an element $X : P(Y)$ to the natural transformation a given by $a_X(f) = P(f)(X)$ for all $f : よ(Y)(X) = \mathbf{C}(X, Y)$. For more details, see [KS06], Proposition 1.4.3. □

Remark 2.14. For any category $\mathbf{D}$, the category $[\mathbf{C}, \mathbf{D}]$ has (co)limits of some kind (terminal or initial object, (co)products, (co)equalizers) if and only if $\mathbf{D}$ does. These (co)limits are computed pointwise: for example, for binary products, $(F \times G)(X) = F(X) \times G(X)$. In particular $P\mathbf{C}$ has all limits and colimits because $\mathbf{Set}$ has all limits and colimits.

Remark 2.15. Also, for any small category $\mathbf{C}$ (small means that the collection of objects is a type in our type universe. For example: for a fixed universe of types, **Type** or **Set** are not small categories, the universe of types is not contained in itself), the presheaf category $P\mathbf{C}$ has exponential objects, given by

$$(F^G)(X) = P\mathbf{C}(ﾖ(X) \times G, F),$$

the natural transformations from the product functor of the Yoneda embedding of X and G, to F (see [MM94], Section I.6, Proposition 1).

Definition 2.16. Suppose that we have a functor $F : \mathbf{C} \to \mathbf{D}$, and $\mathbf{C}$ and $\mathbf{D}$ both have binary products. Then for $X, Y : \mathbf{C}$, we have morphisms $\pi_1 : X \times Y \to X$ and $\pi_2 : X \times Y \to Y$ and applying F to these yields morphisms from $F(X \times Y)$ to $F(X)$ and $F(Y)$. We then have a product morphism

$$\langle F(\pi_1), F(\pi_2)\rangle : F(X \times Y) \to F(X) \times F(Y).$$

Now, if this morphism is an isomorphism for all $X, Y : \mathbf{C}$, we say that F *preserves binary products*.

One can imagine that there exists a similar definition of preservation of limits in general, in which the limit morphism of 'F applied to the projections from the limit' is an isomorphism, for any limit in $\mathbf{C}$.

Definition 2.17. Suppose that we have a functor $F : \mathbf{C} \to \mathbf{D}$ that preserves binary products, and $\mathbf{C}$ and $\mathbf{D}$ have exponential objects. Then for $X, Y : \mathbf{C}$, we have natural bijections

$$\varphi : \mathbf{C}(X^Y \times Y, X) \xrightarrow{\sim} \mathbf{C}(X^Y, X^Y) \quad \text{and} \quad \psi : \mathbf{D}(F(X^Y) \times F(Y), F(X)) \xrightarrow{\sim} \mathbf{D}(F(X^Y), F(X)^{F(Y)})$$

This gives a morphism $F(\varphi^{-1}(\mathrm{id}_{X^Y})) : \mathbf{D}(F(X^Y \times Y), F(X))$. Precomposing with the inverse of the isomorphism $f : F(X^Y \times Y) \xrightarrow{\sim} F(X^Y) \times F(Y)$ and applying ψ gives

$$\psi(f^{-1} \cdot \varphi^{-1}(\mathrm{id}_{X^Y})) : \mathbf{D}(F(X^Y), F(X)^{F(Y)}).$$

We say that F *preserves exponentials* if this morphism is an isomorphism for all $X, Y : \mathbf{C}$. A name for a functor that preserves exponentials is a *cartesian functor*.

Lemma 2.18. *The Yoneda embedding preserves limits.*

Proof. See [Bor94], Volume 1, Proposition 2.15.5 for the full proof, but note that for binary products, we have bijections

$$ﾖ(X \times Y)(Z) \xrightarrow{\sim} ﾖ(X)(Z) \times ﾖ(Y)(Z)$$

sending $f : \mathbf{C}(Z, X \times Y)$ to the pair $(f \cdot \pi_1, f \cdot \pi_2)$ and conversely, sending a pair $(g_1, g_2) : \mathbf{C}(Z, X) \times \mathbf{C}(Z, Y)$ to the product morphism $\langle g_1, g_2\rangle : ﾖ(Z, X \times Y)$. This idea is also the core of the proof about general limits. □

Lemma 2.19 (`yoneda_preserves_exponentials`)**.** *The Yoneda embedding preserves exponentials.*

Proof. First of all, note that for $X, Y, Z : \mathbf{C}$, we have a sequence of isomorphisms [Kam]

$$\begin{aligned}
ﾖ(X^Y)(Z) &\cong P\mathbf{C}(ﾖ(Z), ﾖ(X^Y)) \\
&\cong \mathbf{C}(Z, X^Y) \\
&\cong \mathbf{C}(Z \times Y, X) \\
&\cong P\mathbf{C}(ﾖ(Z \times Y), ﾖ(X)) \\
&\cong P\mathbf{C}(ﾖ(Z) \times ﾖ(Y), ﾖ(X)) \\
&\cong P\mathbf{C}(ﾖ(Z), ﾖ(X)^{ﾖ}(Y)) \\
&\cong (ﾖ(X)^{ﾖ}(Y))(Z)
\end{aligned}$$

using (once or twice) the Yoneda lemma, fully faithfulness of the Yoneda embedding, the property of the exponential object, and the fact that the Yoneda embedding preserves binary products.

Some calculating shows that when applying this isomorphism to some $f : よ(X^Y)(Z)$, we get the natural transformation $h : P\mathbf{C}(よ(Z) \times よ(Y), よ(X))$ given by

$$h_Z(g_1, g_2) = \langle g_1, g_2 \rangle \cdot \varphi^{-1}(f)$$

for all $Z : \mathbf{C}$ and $(g_1, g_2) : よ(Z)(Z) \times よ(Y)(Z)$ and with the natural bijection

$$\varphi : \mathbf{C}(Z \times Y, X) \xrightarrow{\sim} \mathbf{C}(Z, X^Y).$$

It turns out that when we apply the morphism in Definition 2.17 to f, we get exactly the same natural transformation. Therefore, the morphism in Definition 2.17 is the isomorphism defined here and we conclude that よ preserves exponentials. □

For a functor between categories $f : \mathbf{C} \to \mathbf{D}$, given the Yoneda embeddings $よ_{\mathbf{C}} : \mathbf{C} \to P\mathbf{C}$ and $よ_{\mathbf{D}} : \mathbf{D} \to P\mathbf{D}$ (we will often omit the subscript $\mathbf{C}$ and $\mathbf{D}$), we can create a diagram

$$\begin{array}{ccc} \mathbf{C} & \xrightarrow{f} & \mathbf{D} \\ \downarrow{\scriptstyle よ} & & \downarrow{\scriptstyle よ} \\ P\mathbf{C} & \xleftarrow[f_*^{\mathrm{op}}]{} & P\mathbf{D} \end{array}$$

Note that the arrows in this diagram are functors, so objects in a category, instead of elements of a set. Therefore, it often does not make a lot of sense to talk about 'equality' of the functors along the different paths, but we rather talk about isomorphism in the functor category $[\mathbf{C}, P\mathbf{C}]$. If we have such an isomorphism, we say the diagram '2-commutes':

Lemma 2.20. *If $f : \mathbf{C} \to \mathbf{D}$ is a fully faithful functor, the diagram above 2-commutes.*

Proof. For $Y, X : \mathbf{C}$, since f is fully faithful, we have isomorphisms of sets, given by

$$よ(Y)(X) = \mathbf{C}(X, Y) \xrightarrow[f_{X,Y}]{\sim} \mathbf{D}(f(X), f(Y)) = f_*^{\mathrm{op}}(よ(f(Y)))(X).$$

Also, for $g : \mathbf{C}(X', X)$, the following diagram commutes

$$\begin{array}{ccc} よ(Y)(X) & \xrightarrow{f_{X,Y}} & f_*^{\mathrm{op}}(よ(f(Y)))(X) \\ \downarrow{\scriptstyle g\cdot -} & & \downarrow{\scriptstyle f_{X',X}(g)\cdot -} \\ よ(Y)(X') & \xrightarrow{f_{X',Y}} & f_*^{\mathrm{op}}(よ(f(Y)))(X') \end{array}$$

so the isomorphism is natural in X and we have $よ(Y) \cong f_*^{\mathrm{op}}(よ(f(Y)))$ in $P\mathbf{C}$. Lastly, for $g : \mathbf{C}(Y, Y')$ and $X : \mathbf{C}$, the following diagram commutes

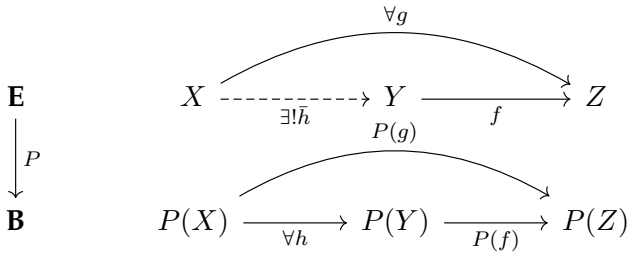
$$\begin{array}{ccc} よ(Y)(X) & \xrightarrow{f_{X,Y}} & f_*^{\mathrm{op}}(よ(f(Y)))(X) \\ \downarrow{\scriptstyle -\cdot g} & & \downarrow{\scriptstyle -\cdot f_{Y,Y'}(g)} \\ よ(Y')(X) & \xrightarrow{f_{X,Y'}} & f_*^{\mathrm{op}}(よ(f(Y')))(X) \end{array}$$

so the isomorphism is natural in Y and we have $よ \cong f \bullet よ \bullet f_*^{\mathrm{op}}$ in $[\mathbf{C}, P\mathbf{C}]$. □

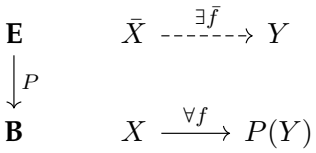
2.5 Fibrations

Let $P : \mathbf{E} \to \mathbf{B}$ be a functor. In this case, we will view this as the category $\mathbf{E}$ 'lying over' the category $\mathbf{B}$, with for every point $X : \mathbf{B}$, a slice $\mathbf{E}_X = P^{-1}(\mathbf{B})$ lying 'above' X.

Definition 2.21. A morphism $f : \mathbf{E}(Y, Z)$ is called *cartesian* if for all $g : \mathbf{E}(X, Z)$ and $h : \mathbf{B}(P(X), P(Y))$ with $h \cdot P(f) = P(g)$, there exists $\bar{h} : \mathbf{E}(X, Y)$ such that $P(\bar{h}) = h$ and $\bar{h} \cdot f = g$, like illustrated in the following diagram from [nLa24b]



Definition 2.22. P is a *fibration* if for all $Y : \mathbf{E}$ and morphisms $f : \mathbf{B}(X, P(Y))$, there exist an object $\bar{X} : \mathbf{E}$ and a cartesian morphism $\bar{f} : \mathbf{E}(\bar{X}, Y)$ such that $P(\bar{X}) = X$ and $P(\bar{f}) = f$:



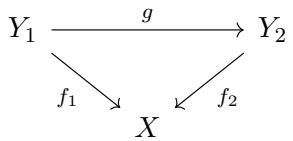
2.6 (Co)slice Categories

Given an object in a category $X : \mathbf{C}$, the morphisms to and from X constitute the slice and coslice categories

Definition 2.23. The *slice category* $\mathbf{C} \downarrow X$ is the category with as objects the morphisms to X:

$$(\mathbf{C} \downarrow X)_0 = \sum_{Y:\mathbf{C}} \mathbf{C}(Y, X).$$

The morphisms from (Y_1, f_1) to (Y_2, f_2) are the morphisms $g : Y_1 \to Y_2$ making the following diagram commute.



The *coslice category* $X \downarrow \mathbf{C}$ is similar, but with the morphisms *from* X instead of *to* X:

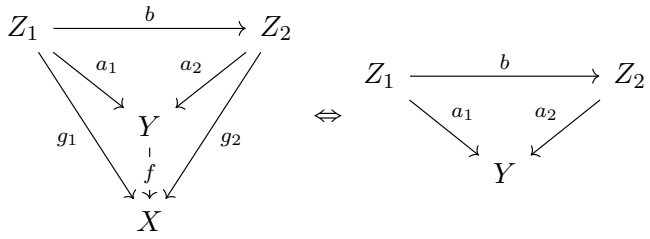
$$(X \downarrow \mathbf{C})_0 = \sum_{Y:\mathbf{C}} \mathbf{C}(X, Y).$$

Now, if we have an object in a slice category, we can again look at the slice of the slice category over that object. However, this gives us nothing new:

Lemma 2.24. *Let (Y, f) be an object in the slice category $(\mathbf{C} \downarrow X)$. The slice category $((\mathbf{C} \downarrow X) \downarrow (Y, f))$ is equivalent to $(\mathbf{C} \downarrow Y)$.*

Proof. An object of $((Z, g), a) : ((\mathbf{C} \downarrow X) \downarrow (Y, f))$ is an object $(Z, g) : (\mathbf{C} \downarrow X)$, together with a morphism $a : (\mathbf{C} \downarrow X)((Z, g), (Y, f))$. That is, a morphism $a : \mathbf{C}(Z, Y)$ such that the obvious triangle commutes (shown in the diagram below on the far left).

Then a morphism between $((Z_1, g_1), a_1)$ and $((Z_2, g_2), a_2)$ is a morphism b between (Z_1, g_1) and (Z_2, g_2) that commutes with a_1 and a_2. Note that a morphism between (Z_1, g_1) and (Z_2, g_2) is a morphism between Z_1 and Z_2 that commutes with g_1 and g_2.



Now, note that g_1 and g_2 are completely determined by $g_i = a_i \cdot f$, so we can leave them out. Also, if b commutes with a_1 and a_2, it automatically also commutes with g_1 and g_2. Therefore, as shown above, we have a correspondence between objects and morphisms

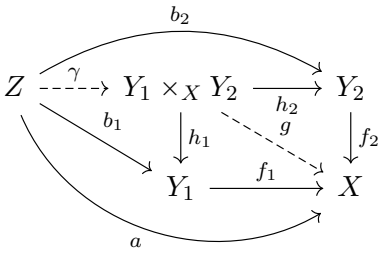
$$b : ((Z_1, g_1), a_1) \to ((Z_2, g_2), a_2) \Leftrightarrow b : (Z_1, a_1) \to (Z_2, a_2).$$

□

Now, the slice categories inherit some structure from the original category:

Lemma 2.25. *For $(Y_1, f_1), (Y_2, f_2) : (\mathbf{C} \downarrow X)$, their product in the slice category is given by their pullback, or fibered product, $Y_1 \times_X Y_2$, together with the induced morphism $g : Y_1 \times_X Y_2 \dashrightarrow X$.*

Proof. Consider the diagram below. The fibered product gives 'projections' h_1 and h_2. Also, if we have some $(Z, a) : (\mathbf{C} \downarrow X)$, together with morphisms $b_1 : (Z, a) \to (Y_1, f_1)$ and $b_2 : (Z, a) \to (Y_2, f_2)$. Then b_1 and b_2 commute with f_1 and f_2, so by the universal property of the fibred product, there exists a unique morphism $\gamma : Z \to Y_1 \times_X Y_2$ that makes the triangles with b_1 and h_1, and with b_2 and h_2 commute. Then γ commutes with a and g as well, so it is a morphism in $(\mathbf{C} \downarrow X)$. This shows that $(Y_1 \times_X Y_2, g)$ has the universal property of the product in $(\mathbf{C} \downarrow X)$.



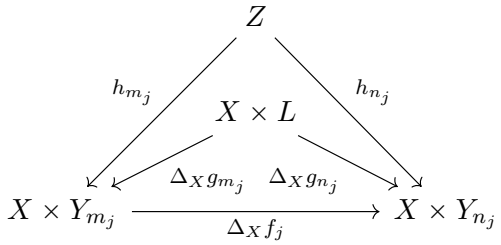
□

For a category $\mathbf{C}$ with products, Hyland introduces the notation $\Delta_X Y$ for the element $(X \times Y, p_1) : (\mathbf{C} \downarrow X)$, and we will follow his example in this.

In fact, $\Delta_X : \mathbf{C} \to (\mathbf{C} \downarrow X)$ is a functor, with $\Delta_X(f) = \mathrm{id}_X \times f$ for $f : \mathbf{C}(Y, Y')$. This functor preserves the terminal object, products and pullbacks:

Lemma 2.26. *Δ_X preserves all limits.*

Proof. Take a diagram $((Y_i)_i, (f_j)_j)$ in $\mathbf{C}$. Suppose that this has a limit $L : \mathbf{C}$ with projections $g_i : \mathbf{C}(L, Y_i)$. Now, consider an object $(Z, q) : (\mathbf{C} \downarrow X)$, together with morphisms $h_i : (\mathbf{C} \downarrow X)((Z, q), \Delta_X Y_i)$, that commute with the $\Delta_X f_j$:



Then the morphisms in $(\mathbf{C} \downarrow X)((Z,q), \Delta_X L)$, commuting with the $\Delta_X g_j$ and h_j are the morphisms in $\mathbf{C}(Z, X \times L) \cong \mathbf{C}(Z,X) \times \mathbf{C}(Z,L)$ that commute with the projections to X and the $\mathrm{id}_X \times g_j$ and h_j. Since the morphisms in $\mathbf{C}(Z,X)$ commuting with q and id_X are exactly q, we can forget about this component, and the morphisms we are looking for correspond to the morphisms in $\mathbf{C}(Z,L)$ that commute with the g_j and $h_j \cdot p_1$. Since $(L, (g_j)_j)$ is a limit, this is a unique morphism. □

Lemma 2.27. *Δ_X preserves exponential objects:*

Proof. See [Bor94], Volume 3, Lemma 5.8.2. This lemma shows this via the following equivalences:

$$\begin{aligned}(\mathbf{C} \downarrow X)((W,f), \Delta_X Y^Z) &\cong \mathbf{C}(W, Y^Z) \\ &\cong \mathbf{C}(W \times Z, Y) \\ &\cong (\mathbf{C} \downarrow X)((W \times (X \times Z), \langle f, p_1 \rangle), \Delta_X Y) \\ &\cong (\mathbf{C} \downarrow X)((W,f) \times \Delta_X Z, \Delta_X Y)\end{aligned}$$

□

2.7 Dependent Products and Sums

The following is based loosely on Section 4.1 of [Tay86].

Take a category $\mathbf{C}$. To talk about dependent sums $\sum_{x:X} Y_x$ and products $\prod_{x:X} Y_x$ in $\mathbf{C}$, we first need some way to construct the family of objects $(Y_x)_x$. Of course, we can do this *externally* using a set X, and picking an object $Y_x : \mathbf{C}$ for every element $x : X$. We then have a category of such families $\mathbf{C}^X$, with objects $(Y_x)_x$ and morphisms $(f_x)_x : \mathbf{C}^X((Y_x)_x, (Z_x)_x)$, with $f_x : Y_x \to Z_x$. We write $\mathbf{C}^X$ because we can view this as just the X-fold power of $\mathbf{C}$. Now, this assignment of categories $X \mapsto \mathbf{C}^X$ can be turned into a contravariant (pseudo)functor of 2-categories $\mathbf{Set}^{\mathrm{op}} \to \mathbf{Cat}$. It sends a morphism $f : X_1 \to X_2$ to the 'relabeling' or 'substitution' functor $\mathbf{C}^{X_2} \to \mathbf{C}^{X_1}$, $(Y_x)_x \mapsto (Y_{f(x)})_x$.

However, there is also an *internal* representation, as a morphism $Y \to X$. We can turn the collection of these morphisms over all the $X : \mathbf{C}$ simultaneously into a category $\mathbf{C}^2$ (abusing notation a bit, writing 2 for the two-point category $\bullet \to \bullet$). Then taking codomains $(Y \to X) \mapsto X$ gives a functor $\mathbf{C}^2 \to \mathbf{C}$. The fiber of this functor above X is the slice category $(\mathbf{C} \downarrow X)$.

In **Set**, the external and internal ways of indexing are actually equivalent, because given a family $(Y_x)_x$ we can construct a morphism $f = \pi_1 : \sum_{x:X} Y_x \to X$ and conversely, we can recover the family $(Y_x)_x$ as $(f^{-1}(x))_x$.

Also note that for **Set**, if we consider an indexed family $(Z_y)_y$ as some function $Z : Y \to \mathbf{Set}$, then substitution over $f : X \to Y$ is just given by postcomposition $Z \circ f : X \to \mathbf{Set}$. It turns out that in the internal representation this is a pullback

$$\begin{array}{ccc} \sum_{x:X} Z_{f(x)} & \longrightarrow & \sum_{y:Y} Z_y \\ \downarrow{\scriptstyle \pi_1} & \lrcorner & \downarrow{\scriptstyle \pi_1} \\ X & \xrightarrow{\quad f \quad} & Y \end{array}$$

This can be extended to a pullback or 'substitution' functor $f^* : (\mathbf{Set} \downarrow Y) \to (\mathbf{Set} \downarrow X)$, which Taylor calls $\mathtt{P}f$. This turns the functor $\mathbf{Set}^2 \to \mathbf{Set}$ into a fibration. We can construct such a functor f^* for any category $\mathbf{C}$ with pullbacks and any morphism $f : \mathbf{C}(X, Y)$, and this makes the functor $\mathbf{C}^2 \to \mathbf{C}$ into a fibration.

Now, in **Set**, for a family $(Y_x)_x$, consider the dependent product $\prod_{x:X} Y_x$. Its elements $(y_x)_x$ can be identified with morphisms from the terminal set: $\{\star\} \to \prod_{x:X} Y_x$, sending $\star$ to $(y_x)_x$. However, they can also be identified with the morphisms $f : X \to \sum_{x:X} Y_x$ that make the following diagram commute, sending x to y_x:

$$\begin{array}{ccc} X & \xrightarrow{\quad f \quad} & \sum_{x:X} Y_x \\ & {\scriptstyle \mathrm{id}_X} \searrow \quad \swarrow {\scriptstyle \pi_1} & \\ & X & \end{array}$$

These are morphisms in $(\mathbf{Set} \downarrow X)$ from (X, id_X) to $(\sum_{x:X} Y_x, \pi_1)$. Note that for the terminal morphism $f : X \to \{\star\}$, we have $(X, \mathrm{id}_X) = f^*(\{\star\}, \mathrm{id}_{\{\star\}})$. To summarize, we have an equivalence

$$(\mathbf{Set} \downarrow X)\left(f^*(\{\star\}, \mathrm{id}_{\{\star\}}), (Y_x)_x\right) \simeq (\mathbf{Set} \downarrow \{\star\})\left((\{\star\}, \mathrm{id}_{\{\star\}}), \left(\prod_{x:X} Y_x, !\right)\right)$$

for ! the terminal morphism. Now, for an internal indexed family $f : Y \to X$ and given a family of families $((Z_y)_{y:Y_x})_{x:X}$, we can wonder whether we can construct the family of dependent products $(\prod_{y:Y_x} Z_y)_x$. In **Set**, this is definitely possible, and from this, we get an equivalence again

$$\left(\mathbf{Set} \downarrow \sum_{x:X} Y_x\right)(f^*(X, \mathrm{id}_X), (Z_y)_{x,y}) \simeq (\mathbf{Set} \downarrow X)\left((X, \mathrm{id}_X), \left(\prod_{y:Y_x} Z_y\right)_x\right).$$

with $p : \sum_{x:X} \sum_{y:Y_x} Z_y \to \sum_{x:X} Y_x$ defined as $p(x, y, z) = (x, y)$. These equivalences suggest an adjunction $f^* \dashv \prod_{\dots}$. We can use this to define in general

Definition 2.28. For a category $\mathbf{C}$ and a morphism $f : Y \to X$, the *dependent product* along f is, if it exists, the right adjoint to the pullback functor:

$$(\mathbf{C} \downarrow X) \underset{\prod_f}{\overset{f^*}{\underset{\longleftarrow}{\overset{\longrightarrow}{\bot}}}} (\mathbf{C} \downarrow Y)$$

Remark 2.29. As argued above, we can recover the familiar dependent product $\prod_{x:X} Y_x$ of a family $(Y_x)_x$ as the dependent product $\prod_f (\sum_x Y_x, \pi_1)$ along the terminal morphism $f : X \to I$. Here we use the equivalence between $(\mathbf{C} \downarrow I)$ and $\mathbf{C}$.

Now we turn our attention to dependent sums. In **Set**, let $(Y_x)_x$ and $(Y'_x)_x$ be two families over X and let $((Z_y)_{y:Y_x})_{x:X}$ be a family of families. Let $f : \sum_{x:X} Y_x \to X$ be the internal representation of $(Y_x)_x$. A family of maps $g_x : (\sum_{y:Y_x} Z_y)_x \to Y'_x$ consists of maps $Z_y \to Y'_x$ for all $y : Y_x$, so these are maps $g_y : Z_y \to Y'_{f(y)}$. This gives an equivalence

$$(\mathbf{Set} \downarrow X)\left(\left(\sum_{y:Y_x} Z_y\right)_x, (Y'_x)_x\right) \simeq \left(\mathbf{Set} \downarrow \left(\sum_{x:X} Y_x\right)\right)((Z_y)_y, f^*((Y'_x)_x)).$$

This, again, suggests an adjunction which we will use as a definition.

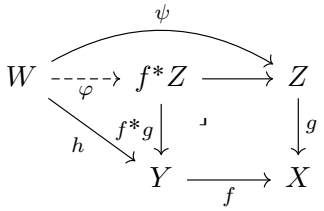
Definition 2.30. For a category $\mathbf{C}$ and a morphism $f : Y \to X$, the *dependent sum* along f is, if it exists, the left adjoint to the pullback functor:

$$(\mathbf{C} \downarrow X) \underset{f^*}{\overset{\Sigma_f}{\underset{\longrightarrow}{\overset{\longleftarrow}{\bot}}}} (\mathbf{C} \downarrow Y)$$

However, note that the conversion from an external to an internal representation in **Set** already contained a dependent sum, which is no coincidence. It turns out that in practice, we will never have a hard time obtaining dependent sums:

Lemma 2.31. *Let $f : \mathbf{C}(Y, X)$ be a morphism in a category. If the pullback functor $f^* : (\mathbf{C} \downarrow X) \to (\mathbf{C} \downarrow Y)$ exists, it has a left adjoint given by postcomposition with f.*

Proof. For morphisms $g : Z \to X$, $h : W \to Y$, the universal property of the pullback, with the following diagram



gives an equivalence between morphisms $\varphi : W \to f^*Z$ that commute with h and f^*g, and morphisms $\psi : W \to Z$ that commute with h, g and f. In other words:

$$(\mathbf{C} \downarrow X)(h \cdot f, g) \simeq (\mathbf{C} \downarrow Y)(h, f^*(g)),$$

which shows the adjunction. □

Now, let $g : Y \to X$ be the internal representation of an indexed family $(Y_x)_x$ and let $f : X \to I$ be the terminal projection. We have $\sum_{x:X} Y_x = g \cdot f : Y \to I$. By the equivalence between $(\mathbf{C} \downarrow I)$ and $\mathbf{C}$, we see that the dependent sum of the family $(Y_x)_x$ is exactly Y. Therefore, our attention is mainly focused on the dependent product.

We will close this section with a name for a category that has all dependent products:

Definition 2.32. A *locally cartesian closed* category is a category $\mathbf{C}$ with pullbacks such that each pullback functor f^* has a right adjoint.

Apart from having dependent sums and products, there also is the following theorem that shows the significance of locally cartesian closedness:

Lemma 2.33. *A category $\mathbf{C}$ is locally cartesian closed if and only if $(\mathbf{C} \downarrow X)$ is cartesian closed for each $X : \mathbf{C}$.*

Proof. See the end of Section 1.3 of [Fre72]. □

Remark 2.34. Note that for $X, Y, Z : \mathbf{C}$ and $f : \mathbf{C}(Y, X)$, the following diagram shows that $f^*\Delta_X Z \cong \Delta_Y Z$:

$$\begin{array}{ccccc} Y \times Z & \xrightarrow{f \times \mathrm{id}_Z} & X \times Z & \xrightarrow{p_2} & Z \\ \downarrow{\scriptstyle p_1} & & \downarrow{\scriptstyle p_1} & & \downarrow{\scriptstyle !} \\ Y & \xrightarrow{f} & X & \xrightarrow{!} & I \end{array}$$

Lemma 2.35. *For $Z, X, Y : \mathbf{C}$ and $p_1 : X \times Y \to X$,*

$$\prod_{p_1} \Delta_{X\times Y} Z \cong \Delta_X Z^Y$$

Proof. First of all, note that $(\mathbf{C} \downarrow X \times Y) \cong ((\mathbf{C} \downarrow X) \downarrow \Delta_X Y)$. Also note that the composite morphism $(X \times Y) \times Z \xrightarrow{p_1} X \times Y \xrightarrow{p_1} X$ is the element $\Delta_X(Z \times Y) : (\mathbf{C} \downarrow X)$.

By Proposition 1.34 in [Fre72], $\prod_{p_1} \Delta_{X\times Y} Z$ is given as the following pullback:

$$\begin{array}{ccc} \prod_{p_1} \Delta_{X\times Y} Z & \longrightarrow & (\Delta_X Y \times Z)^{\Delta_X Y} \\ \downarrow & & \downarrow{\scriptstyle (\Delta_X p_1)^{\Delta_X Y}} \\ X & \longrightarrow & (\Delta_X Y)^{\Delta_X Y} \end{array}$$

By Lemma 2.27, Δ_X preserves exponential objects, so the morphism on the right is $\Delta_X p_1^Y$: $(\mathbf{C} \downarrow X)(\Delta_X(Y \times Z)^Y, \Delta_X Y^Y)$. However, we have an isomorphism $(Y \times Z)^Y \cong Y^Y \times Y^Z$, and then the morphism on the right becomes

$$\Delta_X p_1 : (\mathbf{C} \downarrow X)(\Delta_X(Y^Y \times Y^Z), \Delta_X Y^Y)$$

We also have an isomorphism $X \cong \Delta_X I$. Then by Lemma 2.26 and Remark 2.34, the pullback of this diagram is $\Delta_X Z^Y$:

$$\begin{array}{ccc} \Delta_X Z^Y & \longrightarrow & \Delta_X(Y^Y \times Z^Y) \\ \downarrow & & \downarrow{\scriptstyle \Delta_X p_1} \\ \Delta_X I & \longrightarrow & \Delta_X Y^Y \end{array}$$

□

2.8 (Weakly) Terminal Objects

Definition 2.36. If a category has an object I, such that there is a (not necessarily unique) morphism to it from every other object in the category, I is said to be a *weakly terminal object*.

Definition 2.37. Let $\mathbf{C}$ be a category with terminal object I. For an object $X : \mathbf{C}$, a *global element* of X is a morphism $f : \mathbf{C}(I, X)$.

2.9 Kan Extensions

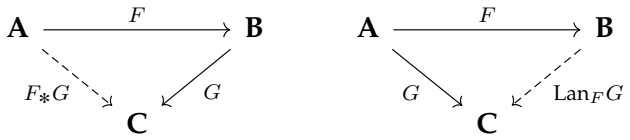
One of the most general and abstract concepts in category theory is the concept of *Kan extensions*. In [Mac98], Section X.7, MacLane notes that

"The notion of Kan extensions subsumes all the other fundamental concepts of category theory."

In this thesis, we will use left Kan extension a handful of times. It comes in handy when we want to extend a functor along another functor in the following way:

Let $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ be categories and let $F : \mathbf{A} \to \mathbf{B}$ be a functor.

Definition 2.38. Precomposition gives a functor between functor categories $F_* : [\mathbf{B}, \mathbf{C}] \to [\mathbf{A}, \mathbf{C}]$. If F_* has a left adjoint, we will denote call this adjoint functor the *left Kan extension* along F and denote it $\mathrm{Lan}_F : [\mathbf{A}, \mathbf{C}] \to [\mathbf{B}, \mathbf{C}]$.



Analogously, when F_* has a right adjoint, one calls this the *right Kan extension* along F and denote it $\mathrm{Ran}_F : [\mathbf{A}, \mathbf{C}] \to [\mathbf{B}, \mathbf{C}]$.

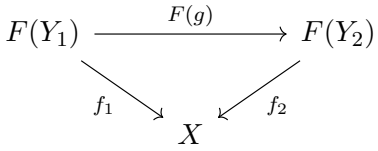
If a category has limits (resp. colimits), we can construct the right (resp. left) Kan extension in a 'pointwise' fashion (see Theorem X.3.1.1 in [Mac98] or Theorem 2.3.3 in [KS06]). Below, I will outline the parts of the construction that we will need explicitly in this thesis.

Lemma 2.39. *If* $\mathbf{C}$ *has colimits,* Lan_F *exists.*

Proof. First of all, for objects $X : \mathbf{B}$, we take

$$(\mathrm{Lan}_F G)(X) := \mathrm{colim}\left((F \downarrow X) \to \mathbf{A} \xrightarrow{G} \mathbf{C}\right).$$

Here, $(F \downarrow X)$ denotes the comma category with as objects the morphisms $\mathbf{B}(F(Y), X)$ for all $Y : \mathbf{A}$, and as morphisms from $f_1 : \mathbf{B}(F(Y_1), X)$ to $f_2 : \mathbf{B}(F(Y_2), X)$ the morphisms $g : \mathbf{A}(Y_1, Y_2)$ that make the diagram commute:



and $(F \downarrow X) \to \mathbf{A}$ denotes the projection functor that sends $f : \mathbf{B}(F(Y), X)$ to Y.

Now, a morphism $h : \mathbf{B}(X_1, X_2)$ gives a morphism of diagrams, sending the $G(Y)$ corresponding to $f : \mathbf{B}(F(Y), X_1)$ to the $G(Y)$ corresponding to $f \cdot h : \mathbf{B}(F(Y), X_2)$. From this, we get a morphism $(\mathrm{Lan}_F G)(h) : \mathbf{C}((\mathrm{Lan}_F G)(X_1), (\mathrm{Lan}_F G)(X_2))$.

The unit of the adjunction is a natural transformation $\eta : \mathrm{id}_{[\mathbf{A},\mathbf{C}]} \Rightarrow \mathrm{Lan}_F \bullet F_*$. We will define this pointwise, for $G : [\mathbf{A}, \mathbf{C}]$ and $Y : \mathbf{A}$. Our diagram contains the $G(Y)$ corresponding to $\mathrm{id}_{F(Y)} : (F \downarrow F(Y))$ and the colimit cocone gives a morphism

$$\eta_G(Y) : \mathbf{C}(G(Y), \mathrm{Lan}_F G(F(Y))),$$

the latter being equal to $(\mathrm{Lan}_F \bullet F_*)(G)(Y)$.

The co-unit of the adjunction is a natural transformation $\epsilon : F_* \bullet \mathrm{Lan}_F \Rightarrow \mathrm{id}_{[\mathbf{B},\mathbf{C}]}$. We will also define this pointwise, for $G : [\mathbf{B}, \mathbf{C}]$ and $X : \mathbf{B}$. The diagram for $\mathrm{Lan}_F(F_* G)(X)$ consists of $G(F(Y))$ for all $f : \mathbf{B}(F(Y), X)$. Then, by the universal property of the colimit, the morphisms $G(f) : \mathbf{C}(G(F(Y)), G(X))$ induce a morphism

$$\epsilon_G(X) : \mathbf{C}(\mathrm{Lan}_F(F_* G)(X), G(X)).$$

□

Lemma 2.40 (`pre_comp_split_essentially_surjective`)**.** *If* $F : \mathbf{A} \to \mathbf{B}$ *is a fully faithful functor, and* $\mathbf{C}$ *is a category with colimits,* $\eta : id_{[\mathbf{A},\mathbf{C}]} \Rightarrow Lan_F \bullet F_*$ *is a natural isomorphism.*

Proof. To show that η is a natural isomorphism, we have to show that $\eta_G(Y) : G(Y) \Rightarrow \mathrm{Lan}_F G(F(Y))$ is an isomorphism for all $G : [\mathbf{A}, \mathbf{C}]$ and $Y : \mathbf{A}$. Since a left adjoint is unique up to natural isomorphism ([AW23], Exercise 153), we can assume that $\mathrm{Lan}_F G(F(Y))$ is given by

$$\mathrm{colim}((F \downarrow F(Y)) \to \mathbf{A} \xrightarrow{G} \mathbf{C}).$$

Now, the diagram for this colimit consists of $G(X)$ for each arrow $f : \mathbf{B}(F(X), F(Y))$. Since F is fully faithful, we have $f = F(\overline{f})$ for some $\overline{f} : \mathbf{A}(X, Y)$. If we now take the arrows $G(\overline{f}) : \mathbf{C}(G(X), G(Y))$, the universal property of the colimit gives an arrow

$$\varphi : \mathbf{C}(\mathrm{Lan}_F G(F(Y)), G(Y))$$

which constitutes an inverse to $\eta_G(Y)$. The proof of this revolves around properties of the colimit and its (induced) morphisms. □

Remark 2.41. In the same way, if $\mathbf{C}$ has limits, ϵ is a natural isomorphism.

Corollary 2.42. *If* $\mathbf{C}$ *has limits or colimits, precomposition of functors* $[\mathbf{B}, \mathbf{C}]$ *along a fully faithful functor is (split) essentially surjective.*

Proof. For each $G : [\mathbf{A}, \mathbf{C}]$ we take $\mathrm{Lan}_F G : [\mathbf{B}, \mathbf{C}]$, and we have $F_*(\mathrm{Lan}_F G) \cong G$. □

Corollary 2.43. *If* $\mathbf{C}$ *has colimits (resp. limits), left (resp. right) Kan extension of functors* $[\mathbf{A}, \mathbf{C}]$ *along a fully faithful functor is fully faithful.*

Proof. Since left Kan extension along F is the left adjoint to precomposition, we have

$$[\mathbf{A}, \mathbf{C}](\mathrm{Lan}_F G, \mathrm{Lan}_F G') \cong [\mathbf{B}, \mathbf{C}](G, F_*(\mathrm{Lan}_F G')) \cong [\mathbf{B}, \mathbf{C}](G, G').$$

□

2.10 Coends

This section is based on Section 1.2 of [Rie14].

In this thesis, we will encounter co-ends a couple of times, so we will introduce them here.

Definition 2.44. Let $\mathbf{C}, \mathbf{D}$ be categories and $F : \mathbf{C}^{\mathrm{op}} \times \mathbf{C} \to \mathbf{D}$ a functor. We define the *coend* $\int^{X:\mathbf{C}} F(X, X)$ to be the colimit

$$\coprod_{f:\mathbf{C}(Y,Z)} F(Z, Y) \underset{F(\mathrm{id}_Y, f)}{\overset{F(f, \mathrm{id}_Z)}{\rightrightarrows}} \coprod_{X:\mathbf{C}} F(X, X) \dashrightarrow \int^{\mathbf{C}} F$$

Remark 2.45. An alternative way to phrase this, is that $\int^{\mathbf{C}} F : \mathbf{D}$ is an object, equipped with arrows $F(X, X) \to \int^{\mathbf{C}} F$ such that for all $f : \mathbf{C}(Y, Z)$, the following diagram must commute

$$\begin{array}{ccc} F(Z, Y) & \xrightarrow{F(f, \mathrm{id}_Y)} & F(Y, Y) \\ \downarrow{\scriptstyle F(\mathrm{id}_Z, f)} & & \downarrow \\ F(Z, Z) & \longrightarrow & \int^{\mathbf{C}} F \end{array}$$

and such that for any other $G : \mathbf{D}$ with the same properties, we have a unique morphism $\int_{\mathbf{C}} F \to G$, making the triangles commute

$$\begin{array}{ccc} F(X, X) & \longrightarrow & \int_{\mathbf{C}} F \\ & \searrow & \downarrow \\ & & G \end{array}$$

Remark 2.46. Of course, a co-end is actually the dual notion of an end, which can be defined as the equalizer of the diagram above, but with the arrows reversed.

Remark 2.47. Left Kan extension can be expressed as a coend:

$$\mathrm{Lan}_F G(Y) = \int^{X:\mathbf{A}} \mathbf{D}(Fa, Y) \cdot GX$$

where $S \cdot Z$ for S a set and $Z : \mathbf{C}$ denotes the 'copower', which intuitively acts as the S-fold coproduct:

$$S \cdot Z = \coprod_{s:S} Z.$$

2.11 Monoids as Categories

Take a monoid M.

Definition 2.48 (`monoid_to_category_ob`)**.** We can construct a category $\mathbf{C}_M$ with $\mathbf{C}_{M0} = \{\star\}$, $\mathbf{C}_M(\star, \star) = M$. The identity morphism on $\star$ is the identity $1 : M$. The composition is given by multiplication $g \cdot_{\mathbf{C}_M} f = f \cdot_M g$.

Remark 2.49 (`monoid_to_category`). Actually, we have a functor from the category of monoids to the category of setcategories (categories whose object type is a set).

A monoid morphism $f : M \to N$ is equivalent to a functor $F_f : \mathbf{C}_M \to \mathbf{C}_N$. Any functor $F_f : \mathbf{C}_M \to \mathbf{C}_N$ sends $\star_M$ to $\star_N$ and corresponds to the monoid morphism as $F_f(m) = f(m)$ for $m : \mathbf{C}_M(\star, \star) = M$.

Lemma 2.50. *An isomorphism of monoids gives an (adjoint) equivalence of categories.*

Proof. Given an isomorphism $f : M \xrightarrow{\sim} N$, we have functors $F_f : \mathbf{C}_M \to \mathbf{C}_N$ and $F_{f^{-1}} : \mathbf{C}_N \to \mathbf{C}_M$. Take the identity natural transformations $\eta : \mathrm{id}_{\mathbf{C}_M} \Rightarrow F_f \bullet F_{f^{-1}}$ and $\epsilon : F_{f^{-1}} \bullet F_f \Rightarrow \mathrm{id}_{\mathbf{C}_N}$. Of course these are natural isomorphisms. □

Definition 2.51 (`monoid_action`)**.** A *right monoid action* of M on a set X is a function $X \times M \to X$ such that for all $x : X$, $m, n : M$,

$$x1 = x \qquad \text{and} \qquad (xm)n = x(m \cdot n).$$

Definition 2.52 (`monoid_action_morphism`)**.** A *morphism* between sets X and Y with a right M-action is an M-equivariant function $f : X \to Y$: a function such that $f(xm) = f(x)m$ for all $x : X$ and $m : M$.

These, together with the identity and composition from **Set**, constitute a category $\mathbf{RAct}_M$ of right M-actions (`monoid_action_cat`).

Lemma 2.53 (`monoid_presheaf_action_equivalence`)**.** *There is an adjoint equivalence between the presheaf category $P\mathbf{C}_M$ and $\mathbf{RAct}_M$.*

Proof. This correspondence sends a presheaf F to the set $F(\star)$, and conversely, the set X to the presheaf F given by $F(\star) := X$. The M-action corresponds to the presheaf acting on morphisms as $xm = F(m)(x)$. A morphism (natural transformation) between presheaves $F \Rightarrow G$ corresponds to a function $F(\star) \to G(\star)$ that is M-equivariant, which is exactly a monoid action morphism. □

Remark 2.54. Since $\mathbf{RAct}_M$ is equivalent to a presheaf category, it has all limits. However, we can make this concrete. The set of the product $\prod_i X_i$ is the product of the underlying sets. The action is given pointwise by $(x_i)_i m = (x_i m)_i$.

Note that the terminal set with M-action is $\{\star\}$, with action $\star m = \star$ (`terminal_monoid_action`).

Lemma 2.55 (`monoid_action_global_element_fixpoint_iso`)**.** *The global elements of $X : \mathbf{RAct}_M$ correspond to $x : X$ that are invariant under the M-action.*

Proof. A global element of X is a morphism $f : \{\star\} \to X$ such that for all $m : M$, $f(\star)m = f(\star m) = f(\star)$. Therefore, it is given precisely by the element $f(\star) : X$, which must be invariant under the M-action. □

Lemma 2.56 (`is_exponentiable_monoid_action`)**.** *The category $\mathbf{RAct}_M$ has exponentials.*

Proof. Given $X, Y : \mathbf{RAct}_M$. Consider the set $\mathbf{C}(M \times X, Y)$ with an M-action given by $(fm')(m, x) = f(m'm, x)$ for $f : \mathbf{C}(M \times X, Y)$. This is the exponential object X^Y, with the (universal) evaluation morphism $X \times X^Y \to Y$ given by $(x, f) \mapsto f(1, x)$. Explicitly, we get a natural isomorphism $\psi : \mathbf{RAct}_M(Z \times Y, X) \xrightarrow{\sim} \mathbf{RAct}_M(Z, X^Y)$ given by

$$\psi(f)(z)(m, y) = f(zm, y) \quad \text{and} \quad \psi^{-1}(g)(z, y) = g(z)(1, y).$$

□

Definition 2.57 (`monoid_monoid_action`)**.** We can view M as a set U_M with right M-action $mn = m \cdot n$ for $m : U_M$ and $n : M$. Note that U_M is the Yoneda embedding of the object $\star : \mathbf{C}_M$.

2.11.1 Extension and restriction of scalars

Let $f : M \to N$ be a morphism of monoids.

Remember that $\mathbf{RAct}_M$ is equivalent to the functor category $P\mathbf{C}_M$. Also, f is equivalent to a functor $F_f : \mathbf{C}_M \to \mathbf{C}_N$. The following is a specific case of the concepts in the section about Kan extension:

Lemma 2.58 (`scalar_restriction_functor`)**.** *We get a restriction of scalars functor $f^* : \mathbf{RAct}_N \to \mathbf{RAct}_M$.*

Proof. Given a set X with right N-action, take the set X again, and give it a right M-action, sending (x, m) to $xf(m)$.

On morphisms, send an N-equivariant morphism $f : X \to Y$ to the M-equivariant morphism $f : X \to Y$. □

Since **Set** has colimits, and restriction of scalars corresponds to precomposition of $\mathbf{C}_N$-presheaves, we can give it a left adjoint. This is the (pointwise) left Kan extension, which boils down to a very concrete definition, reminiscent of a tensor product:

Lemma 2.59 (`scalar_extension_functor`)**.** *We get an extension of scalars functor $f_* : \mathbf{RAct}_M \to \mathbf{RAct}_N$.*

Proof. Given $X : \mathbf{RAct}_M$, take $Y = X \times N/\sim$ with the relation $(xm, n) \sim (x, f(m) \cdot n)$ for $m : M$. This has a right N-action given by $(x, n_1)n_2 = (x, n_1 n_2)$.

On morphisms, it sends the m-equivariant $f : X \to X'$ to the morphism $(x, n) \mapsto (f(x), n)$. □

Lemma 2.60 (`scalar_extension_preserves_monoid_monoid_action`)**.** *For U_M the set M with right M-action, we have $f_*(U_M) \cong U_N$.*

Proof. The proof relies on the fact that for all $m : U_M$ and $n : N$, we have

$$(m, n) \sim (1, f(m)n).$$

□

Consider the category $\mathbf{D}$ with $\mathbf{D}_0 = N$ and

$$\mathbf{D}(n_1, n_2) = \{m : M \mid f(m) \cdot n_1 = n_2\}.$$

Lemma 2.61 (`scalar_extension_preserves_terminal`)**.** *Suppose that* $\mathbf{D}$ *has a weakly terminal element. Then for* $I_M : \mathbf{RAct}_M$ *the terminal object, we have* $f_*(I_M) \cong I_N$.

Proof. If $\mathbf{D}$ has a weakly terminal object, there exists $n_0 : N$ such that for all $n : N$, there exists $m : M$ such that $f(m) \cdot n = n_0$.

The proof then relies on the fact that every element of $f_*(I_M)$ is given by some $(\star, n)$, but then there exists some $m : M$ such that

$$(\star, n) = (\star \cdot m, n) \sim (\star, f(m) \cdot n) = (\star, n_0),$$

so $f_*(I_M)$ has exactly 1 element. □

Remark 2.62. For f_* to preserve terminal objects, we actually only need $\mathbf{D}$ to be connected. The fact that $f_*(I_M)$ is a quotient by a symmetric and transitive relation then allows us to 'walk' from any $(\star, n_1)$ to any other $(\star, n_2)$ in small steps.

For any $n_1, n_2 : N$, consider the category $\mathbf{D}_{n_1,n_2}$, given by

$$\mathbf{D}_{n_1,n_2,0} = \{(n, m_1, m_2) : N \times M \times M \mid n_i = f(m_i) \cdot n\}$$

and

$$\mathbf{D}_{n_1,n_2}((n, m_1, m_2), (\overline{n}, \overline{m}_1, \overline{m}_2)) = \{m : M \mid f(m) \cdot n = \overline{n}, m_i = \overline{m}_i \cdot m\}.$$

Lemma 2.63 (`scalar_extension_preserves_binproducts`)**.** *Suppose that* $\mathbf{D}_{n_1,n_2}$ *has a weakly terminal object for all* $n_1, n_2 : N$*. Then for* $X, Y : \mathbf{RAct}_M$*, we have* $f_*(X \times Y) \cong f_*(X) \times f_*(Y)$.

Proof. Now, any element in $f_*(X) \times f_*(Y) = (X \times N/\sim) \times (Y \times N/\sim)$ is given by some (a, n_1, b, n_2).

The fact that $\mathbf{D}_{n_1,n_2}$ has a weakly terminal object means that we have some $\overline{n} : N$ and $\overline{m}_1, \overline{m}_2 : M$ with $n_i = f(\overline{m}_i) \cdot \overline{n}$. Therefore,

$$(a, n_1, b, n_2) = (a, f(\overline{m}_1) \cdot \overline{n}, b, f(\overline{m}_2) \cdot \overline{n}) \sim (a\overline{m}_1, \overline{n}, b\overline{m}_2, \overline{n}),$$

so this is equivalent to $(a\overline{m}_1, b\overline{m}_2, \overline{n}) : f_*(X \times Y)$. Note that this trivially respects the right N-action.

The fact that $(\overline{n}, \overline{m}_1, \overline{m}_2)$ is weakly terminal also means that for all $n : N$ and $m_1, m_2 : M$ with $n_i = f(m_i) \cdot n$, there exists $m : M$ such that $f(m) \cdot n = \overline{n}$ and $m_i = \overline{m}_i \cdot m$. This means that the equivalence that we established is actually well-defined: equivalent elements in $f_*(X) \times f_*(Y)$ are sent to equivalent elements in $f_*(X \times Y)$.

Therefore, we have an isomorphism $\psi : f^*(X) \times f^*(Y) \xrightarrow{\sim} f^*(X \times Y)$. Now we only need to show that the projections are preserved by this isomorphism. To that end, take $x = (a, n_1, b, n_2) \sim (a\overline{m}_1, \overline{n}, b\overline{m}_2, \overline{n}) : f^*(X) \times f^*(Y)$. We have

$$f^*(\pi_1)(\psi(x)) = (a\overline{m}_1, \overline{n}) = \pi_1'(x).$$

In the same way, $f^*(\pi_2) \circ \psi = \pi_2'$ and this concludes the proof. □

2.12 The Karoubi Envelope

Let $\mathbf{C}$ be a category and $X, Y : \mathbf{C}$ objects. We will denote the type of section-retraction pairs of Y onto X with

$$X \lhd Y := \sum_{r:\mathbf{C}(Y,X)} \sum_{s:\mathbf{C}(X,Y)} s \cdot r = \mathrm{id}_X.$$

Now, note that for $(r, s) : X \lhd Y$, $r \cdot s : \mathbf{C}(Y, Y)$ is an idempotent morphism, since $r \cdot s \cdot r \cdot s = r \cdot s$. We can also wonder whether for an idempotent morphism $f : \mathbf{C}(X, X)$, we can find some $Y : \mathbf{C}$ and some $(r, s) : X \lhd Y$ such that $f = r \cdot s$. If this is the case, we say that the idempotent f *splits*. If f does not split, we can wonder whether we can find an embedding $\iota_{\mathbf{C}} : \mathbf{C} \hookrightarrow \overline{\mathbf{C}}$ into some category $\overline{\mathbf{C}}$ such that the idempotent $\iota_{\mathbf{C}}(f) : \overline{\mathbf{C}}(\iota_{\mathbf{C}}(X), \iota_{\mathbf{C}}(X))$ does split. This is one way to arrive at the *Karoubi envelope*:

Definition 2.64 (`karoubi_envelope`)**.** We define the category $\overline{\mathbf{C}}$. The objects of $\overline{\mathbf{C}}$ are tuples (X, f) with $X : \mathbf{C}$, $f : \mathbf{C}(X, X)$ such that $f \cdot f = f$. The morphisms between (X_1, f_1) and (X_2, f_2) are morphisms $g : \mathbf{C}(X_1, X_2)$ such that $f_1 \cdot g \cdot f_2 = g$. This can be summarized in the following diagram:

$$f_1 \circlearrowright X_1 \xrightarrow{g} X_2 \circlearrowleft f_2$$

The identity morphism on (X, f) is given by f and $\overline{\mathbf{C}}$ inherits morphism composition from $\mathbf{C}$.

This category is called the *Karoubi envelope*, the *idempotent completion*, the *category of retracts*, or the *Cauchy completion* of $\mathbf{C}$.

Remark 2.65. Note that for a morphism $f : \overline{\mathbf{C}}((X, a), (Y, b))$,

$$a \cdot f = a \cdot a \cdot f \cdot b = a \cdot f \cdot b = f$$

and in the same way, $f \cdot b = f$.

Definition 2.66 (`karoubi_envelope_inclusion`)**.** We have an embedding $\iota_{\mathbf{C}} : \mathbf{C} \to \overline{\mathbf{C}}$, sending $X : \mathbf{C}$ to (X, id_X) and $f : \mathbf{C}(X, Y)$ to f.

Lemma 2.67 (`karoubi_envelope_is_retract`)**.** *Every object $X : \overline{\mathbf{C}}$ is a retract of $\iota_{\mathbf{C}}(Y)$ for some $Y : \mathbf{C}$.*

Proof. Note that $X = (Y, a)$ for some $Y : \mathbf{C}$ and an idempotent $a : Y \to Y$. We have

$$(a, a) : (Y, a) \lhd \iota_{\mathbf{C}}(Y),$$

since $a \cdot a = a = \mathrm{id}_X$, so X is a retract of $\iota_{\mathbf{C}}(Y)$. □

Lemma 2.68 (`karoubi_envelope_idempotent_splits`)**.** *In $\overline{\mathbf{C}}$, every idempotent splits.*

Proof. Take an idempotent $f : \overline{\mathbf{C}}(X, X)$. Note that X is given by an object $Y : \mathbf{C}$ and an idempotent $a : \mathbf{C}(Y, Y)$. Also, f is given by some idempotent $f : \mathbf{C}(Y, Y)$ with $a \cdot f \cdot a = f$.

Now, we have $(Y, f) : \overline{\mathbf{C}}$ and

$$(f, f) : (Y, f) \lhd X,$$

because $f \cdot f = f = \mathrm{id}_{(Y,f)}$. Also, $f = f \cdot f$, so f is split. □

Lemma 2.69 (`karoubi_envelope_inclusion_fully_faithful`)**.** *$\iota_{\mathbf{C}}$ is fully faithful.*

Proof. This follows immediately from the fact that

$$\overline{\mathbf{C}}((X, \mathrm{id}_X), (Y, \mathrm{id}_Y)) = \{f : \mathbf{C}(X, Y) \mid \mathrm{id}_X \cdot f \cdot \mathrm{id}_Y = f\} = \mathbf{C}(X, Y).$$

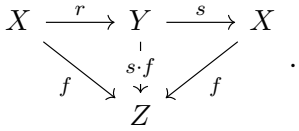
□

Lemma 2.70 (`retract_functor_is_equalizer, retract_functor_is_coequalizer`)**.** *Let* $\mathbf{D}$ *be a category and suppose that we have two objects* $X, Y : \mathbf{D}$ *and a retraction*

$$(r, s) : Y \lhd X.$$

Then Y *is the equalizer of* $X \underset{r \cdot s}{\overset{id_X}{\rightrightarrows}} X$.

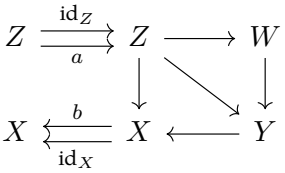
Proof. Suppose that we have an object $Z : \mathbf{D}$ and a morphism f with $(r \cdot s) \cdot f = f$. Then f factors as $r \cdot (s \cdot f)$. Also, for any $g : \mathbf{D}(Y, Z)$ with $r \cdot g = f$, we have $g = s \cdot r \cdot g = s \cdot f$:



In a similar way, Y is also the coequalizer of the given diagram.

Now, note that if we have a coequalizer W of id_Z and a, and an equalizer Y of id_X and b (in particular, if W and Y are retracts), the universal properties of these give an equivalence

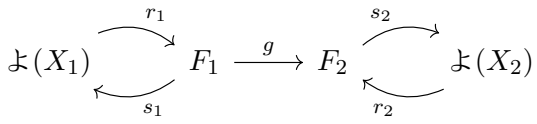
$$\mathbf{D}(W, Y) \cong \{f : \mathbf{D}(Z, Y) \mid a \cdot f = f\} \cong \{f : \mathbf{D}(Z, X) \mid a \cdot f = f = f \cdot b\}.$$



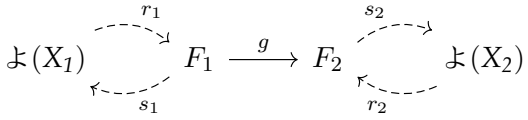
□

Besides $\overline{\mathbf{C}}$, there is also another (classically equivalent) definition of the Karoubi envelope. An object in this alternative category is a presheaf $F : P\mathbf{C}$ that is a retract of the Yoneda embedding $よ(X)$ of an object $X : \mathbf{C}$. Note that there are two different ways to translate this to univalent foundations. We can either interpret the existence of the object X and the retraction $(r, s) : F \lhd よ(X)$ as additional *structure* on F, or we can treat it as a *property* and ask for *mere existence* (see Definition 3.8) of X, r and s. This gives rise to two different categories:

Definition 2.71 (`karoubi_envelope'`)**.** We define the category $\tilde{\mathbf{C}}$ in which every object is a presheaf $F : P\mathbf{C}$, together with an object $X : \mathbf{C}$ and a retraction-section pair $(r, s) : F \lhd よ(X)$. The morphisms from (F_1, X_1, r_1, s_1) to (F_2, X_2, r_2, s_2) are just the presheaf morphisms $f : P\mathbf{C}(F_1, F_2)$. This can be summarized in the following diagram:



Definition 2.72. We define the category $\hat{\mathbf{C}}$ as the full subcategory of $P\mathbf{C}$ consisting of objects $F : P\mathbf{C}$ such that there merely exist an object $X : \mathbf{C}$ and a retraction-section pair $(r, s) : F \lhd よ(X)$, summarized in the following diagram:



This gives us what we need to translate the classical equivalence between the two notions of Karoubi envelope to type theory:

Lemma 2.73 (`karoubi_envelope_is_retract`)**.** *We have an adjoint equivalence* $\overline{\mathbf{C}} \xrightarrow{\sim} \tilde{\mathbf{C}}$.

Proof. As shown in Remark 2.70, an object $(X, f) : \overline{\mathbf{C}}$ is the equalizer of id_X and f. Therefore, we send it to the equalizer

$$F \xrightarrow{s} よ(X) \underset{よ(f)}{\overset{\mathrm{id}_{よ(X)}}{\rightrightarrows}} よ(X)$$

Note that for $よ(f)$, we have $よ(f) \cdot よ(f) = よ(f) \cdot \mathrm{id}_{よ(X)}$, so the universal property of the equalizer gives a morphism $r : P\mathbf{C}(よ(X), F)$ such that $r \cdot s = よ(f)$. Using this same universal property, we can show that $s \cdot r = \mathrm{id}_F$. We send a morphism $g : \overline{\mathbf{C}}((X_1, f_1), (X_2, f_2))$ to

$$s_1 \cdot よ(g) \cdot r_2 : P\mathbf{C}(F_1, F_2).$$

Note that this is an equivalence on the morphisms: For any morphism $\bar{g} : P\mathbf{C}(F_1, F_2)$, we have

$$r_1 \cdot \bar{g} \cdot s_2 : P\mathbf{C}(よ(X_1), よ(X_2)).$$

By the fully faithfulness of the Yoneda lemma, this corresponds to a morphism $g : \mathbf{C}(X_1, X_2)$ and we can show that these two maps between $\mathbf{C}((X_1, f_1), (X_2, f_2))$ and $\overline{\mathbf{C}}(F_1, F_2)$ are inverses of each other.

$$\begin{array}{ccccc} F_1 & \underset{r_1}{\overset{s_1}{\rightleftarrows}} & よ(X_1) & \underset{よ(f_1)}{\overset{\mathrm{id}_{よ(X_1)}}{\rightrightarrows}} & よ(X_1) \\ \downarrow{\scriptstyle \bar{g}} & & \downarrow{\scriptstyle よ(g)} & & \\ F_2 & \underset{r_2}{\overset{s_2}{\rightleftarrows}} & よ(X_2) & \underset{よ(f_2)}{\overset{\mathrm{id}_{よ(X_2)}}{\rightrightarrows}} & よ(X_2) \end{array}$$

Now, note that this map is also split essentially surjective. For some $(F, X, r, s) : \tilde{\mathbf{C}}$, $r \cdot s$ is an idempotent morphism on $よ(X)$, and by fully faithfulness of the Yoneda Lemma, it corresponds to an idempotent morphism f on X. We send (F, X, r, s) to (X, f). Note that both F and the image of (X, f) are equalizers of $\mathrm{id}_{よ(X)}$ and $r \cdot s$, so they are isomorphic.

The fact that a fully faithful and split essentially surjective functor is an adjoint equivalence concludes the proof. □

Remark 2.74. Note that since the morphism types of $\tilde{\mathbf{C}}$ and $\hat{\mathbf{C}}$ are the same, and since the objects of $\hat{\mathbf{C}}$ are just truncated versions of the objects in $\tilde{\mathbf{C}}$, we have a fully faithful embedding

$$\tilde{\mathbf{C}} \hookrightarrow \hat{\mathbf{C}},$$

which just forgets the choices for X, r and s on the objects. Note that this is also essentially surjective: by definition, for any $(F, H) : \hat{\mathbf{C}}$, there merely exist X, r and s such that (X, r, s) truncates to H. Therefore, we have a weak equivalence from $\tilde{\mathbf{C}}$ to $\hat{\mathbf{C}}$. However, as we will see, $\tilde{\mathbf{C}}$ is usually not univalent, so this does not give an adjoint equivalence of categories.

This all leads up to:

Corollary 2.75. *We have three candidates for the category of retracts, which are related to each other and to* $\mathbf{C}$ *and* $P\mathbf{C}$ *as follows:*

$$\mathbf{C} \xhookrightarrow{\iota_{\mathbf{C}}} \overline{\mathbf{C}} \xrightarrow{\sim} \tilde{\mathbf{C}} \xhookrightarrow{} \hat{\mathbf{C}} \quad \subseteq \quad P\mathbf{C}$$

The fact that $\hat{\mathbf{C}}$ is univalent, and has weak equivalences from $\overline{\mathbf{C}}$ and $\tilde{\mathbf{C}}$ exhibits $\hat{\mathbf{C}}$ as the Rezk completion of $\overline{\mathbf{C}}$ and $\tilde{\mathbf{C}}$.

Even though univalence and the Rezk completion in general will be covered in Section 3.2, we will study univalence of the Karoubi envelope here:

Lemma 2.76 (`karoubi_univalence`)**.** *If $\overline{\mathbf{C}}$ is univalent, $\mathbf{C}$ is univalent as well.*

Proof. The fully faithful embedding $\iota_{\mathbf{C}} : \mathbf{C} \hookrightarrow \overline{\mathbf{C}}$ induces equivalences $(X \cong Y) \simeq (\iota_{\mathbf{C}}(X) \cong \iota_{\mathbf{C}}(Y))$. We also have an equivalence $(X = Y) \simeq (\iota_{\mathbf{C}}(X) = \iota_{\mathbf{C}}(Y))$, because, as it turns out, any equality between X and Y also preserves the identity of X. Therefore, if $\overline{\mathbf{C}}$ is univalent, we have a chain of equivalences

$$(X = Y) \simeq (\iota_{\mathbf{C}}(X) = \iota_{\mathbf{C}}(Y)) \simeq (\iota_{\mathbf{C}}(X) \cong \iota_{\mathbf{C}}(Y)) \simeq (X \cong Y),$$

which shows that $\mathbf{C}$ is univalent as well. □

Remark 2.77. Note that the converse does not necessarily hold. Consider the commutative monoid consisting of the three matrices

$$a = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad \text{and} \quad c = \begin{pmatrix} -1 & 0 \\ 0 & 0 \end{pmatrix}$$

under matrix multiplication. As we saw in Section 2.11, we can turn this into a category $\mathbf{C}_M$ with one object $\star$, and the three morphisms a, b and c. Only a is an isomorphism and since $\star = \star$ has exactly one inhabitant, $\mathbf{C}_M$ is univalent.

When we construct the Karoubi envelope $\overline{\mathbf{C}}_M$, we get a category with objects a and b. Now, note that since M is a set, $b = b$ still has one inhabitant. Of course, $b \cong b$ contains the identity automorphism. However, since

$$b \cdot c = c \quad \text{and} \quad c \cdot c = b,$$

c is also an automorphism of b. Therefore, $\overline{\mathbf{C}}_M$ is not univalent.

In a somewhat more complicated way, we can show that $\tilde{\mathbf{C}}_M$ is not univalent either: the counterpart of b in that category has two automorphisms, corresponding to b and c, but only one equality, which is the identity. This is because isomorphisms only have to respect the presheaf structure, whereas equalities also have to respect the section (and retraction). Note that presheaves are univalent, so equality of presheaves is equivalent to isomorphism of presheaves.

Therefore, $\overline{\mathbf{C}}$ and $\tilde{\mathbf{C}}$ are usually more pathological than $\mathbf{C}$ itself. On the other hand, since $\hat{\mathbf{C}}$ is a full subcategory of $P\mathbf{C}$, it is univalent.

Remark 2.78. As we saw in this section, there are multiple candidates for the 'Karoubi envelope'. Two of these candidates are equivalent to each other, but the third is not. Therefore, we need to be careful which definition we choose, because this choice has consequences. On one hand, $\hat{\mathbf{C}}$ is univalent, but very abstract. In this category, there are classical constructions that we cannot do, because for every object, we only have mere existence of an idempotent morphism, and because the objects of $\hat{\mathbf{C}}$ do not form a set, we cannot use the axiom of choice to pick an idempotent morphism. On the other hand, $\overline{\mathbf{C}}$ and $\tilde{\mathbf{C}}$ are very elementary and concrete, which sometimes helps when doing constructions. However, they are not univalent, which makes working with them complicated in a different way.

The remainder of this section works towards the adjoint equivalence between $P\mathbf{C}$ and $P\overline{\mathbf{C}}$.

Since a functor preserves retracts, and since every object of $\overline{\mathbf{C}}$ is a retract of an object in $\mathbf{C}$, we can generalize the construction of the functor that we do in Lemma 2.73 for the functor $よ : \mathbf{C} \to P\mathbf{C}$, to general functors $F : \mathbf{C} \to \mathbf{D}$, if $\mathbf{D}$ has (co)equalizers.

For convenience, the lemma below works very abstractly with pointwise left Kan extension using colimits, but one could also prove this using just (co)equalizers (or right Kan extension using limits).

Lemma 2.79 (`karoubi_pullback_equivalence`)**.** *Let* $\mathbf{D}$ *be a category with colimits. We have an adjoint equivalence between* $[\mathbf{C}, \mathbf{D}]$ *and* $[\overline{\mathbf{C}}, \mathbf{D}]$*.*

Proof. We already have an adjunction $\mathrm{Lan}_{\iota_\mathbf{C}} \dashv \iota_{\mathbf{C}*}$. Also, since $\iota_\mathbf{C}$ is fully faithful, we know that η is a natural isomorphism. Therefore, we only have to show that ϵ is a natural isomorphism. That is, we need to show that $\epsilon_G(X, a) : \mathbf{D}(\mathrm{Lan}_{\iota_\mathbf{C}}(\iota_{\mathbf{C}*}G)(X, a), G(X, a))$ is an isomorphism for all $G : [\overline{\mathbf{C}}, \mathbf{D}]$ and $(X, a) : \overline{\mathbf{C}}$.

One of the components in the diagram of $\mathrm{Lan}_{\iota_\mathbf{C}}(\iota_{\mathbf{C}*}G)(X, a)$ is the $G(\iota_\mathbf{C}(X))$ corresponding to $a : \overline{\mathbf{C}}(\iota_\mathbf{C}(X), (X, a))$. This component has a morphism into our colimit

$$\varphi : \mathbf{C}(G(\iota_\mathbf{C}(X)), \mathrm{Lan}_{\iota_\mathbf{C}}(\iota_{\mathbf{C}*}G)(X, a)).$$

Note that we can also view a as a morphism $a : \overline{\mathbf{C}}((X, a), \iota_\mathbf{C}(X))$. This gives us our inverse morphism

$$G(a) \cdot \varphi : \mathbf{C}(G(X, a), \mathrm{Lan}_{\iota_\mathbf{C}}(\iota_{\mathbf{C}*}G)(X, a)).$$

□

Lemma 2.80 (`opp_karoubi`)**.** *The formation of the opposite category commutes with the formation of the Karoubi envelope.*

Proof. An object in $\overline{\mathbf{C}^{\mathrm{op}}}$ is an object $X : \mathbf{C}^{\mathrm{op}}$ (which is just an object $X : \mathbf{C}$), together with an idempotent morphism $a : \mathbf{C}^{\mathrm{op}}(X, X) = \mathbf{C}(X, X)$. This is the same as an object in $\overline{\mathbf{C}}^{\mathrm{op}}$.

A morphism in $\overline{\mathbf{C}^{\mathrm{op}}}((X, a), (Y, b))$ is a morphism $f : \mathbf{C}^{\mathrm{op}}(X, Y) = \mathbf{C}(Y, X)$ such that

$$b \cdot_\mathbf{C} f \cdot_\mathbf{C} a = a \cdot_{\mathbf{C}^{\mathrm{op}}} f \cdot_{\mathbf{C}^{\mathrm{op}}} b = f.$$

A morphism in $\overline{\mathbf{C}}^{\mathrm{op}}((X, a), (Y, b)) = \overline{\mathbf{C}}((Y, b), (X, a))$ is a morphism $f : \mathbf{C}(Y, X)$ such that $b \cdot f \cdot a = f$.

Now, in both categories, the identity morphism on (X, a) is given by a.

Lastly, $\overline{\mathbf{C}^{\mathrm{op}}}$ inherits morphism composition from $\mathbf{C}^{\mathrm{op}}$, which is the opposite of composition in $\mathbf{C}$. On the other hand, composition in $\overline{\mathbf{C}}^{\mathrm{op}}$ is the opposite of composition in $\overline{\mathbf{C}}$, which inherits composition from $\mathbf{C}$. □

Corollary 2.81. *As the category* **Set** *is cocomplete, we have an equivalence between the category of presheaves on* $\mathbf{C}$ *and the category of presheaves on* $\overline{\mathbf{C}}$*:*

$$[\mathbf{C}^{op}, \mathbf{Set}] \cong [\overline{\mathbf{C}^{op}}, \mathbf{Set}] \cong [\overline{\mathbf{C}}^{op}, \mathbf{Set}].$$

Chapter 3

Univalent Foundations

3.1 Dependent Type Theory

Univalent foundations takes place in a framework of type theory. In this section, we will quickly introduce some topics that we will need in subsequent sections.

First of all, *type theory* is the study of 'type systems'. A *type system* is a collection of *terms* or *elements*, each of which has a corresponding type. A type is much like a set in set theory in that it has elements (for example, true : Bool or $1 : \mathbb{N}$), but there are important differences. First of all, in type theory, terms are declared together with their type, so every term has exactly one type, whereas in set theory, something can be an element of multiple sets, like $5 \in \mathbb{N}$ and $5 \in \mathbb{R}$. Secondly, equality in type theory can work a bit differently than in set-based mathematics, but I will cover this in the next section.

Of course, in computer science, we are very familiar with type systems. In most programming languages, values explicitly (e.g. in Java) or implicitly (e.g. in Python) have a type associated to them. For example, `"Hello, World"` is of type *string*, `true` and `false` are of type *boolean*, `1` is of type *integer* and `-5.8` is of type *floating point number*. And here, already, we see some subtleties: due to their different internal representations, many programming languages distinguish between integers and floating point numbers, even though every integer can be considered to be a floating point number. Usually, programming languages resolve this by offering methods (sometimes in the form of *coercions*) to convert between the two types (discarding what comes after the decimal point when converting a floating point number to an integer). Another coercion that occurs sometimes is the conversion of a *character* like `a`, `1` or `&` to an integer and back.

So if we have types and elements, are types also elements of some type? This is a tricky question, because if we say that there exists a type **Type** which contains all types (and therefore, itself), we introduce inconsistency in our type system (see [Hur95]). This is just like having a 'set of all sets', which results in problematic questions like "does its subset, containing only the sets which do not contain itself, contain itself?" This is usually solved by either stating that types do not themselves have a type, or by assuming the existence of *type universes* $U_1, U_2, \ldots$, such that every type is an element of some U_i. Note that universes are allowed to be inclusive or not: we may have $U_n \subseteq U_{n+1}$. In this thesis, we will not explicitly mention particular universes. We will just write **Type** to denote some type universe (or its category of types and functions). This is called *typical ambiguity*, meaning that every theorem holds for all (suitable) assignments of universes to these instances of **Type**.

One notable class of types is given by the *function types*. For types A and B, our type system might have the type $A \to B$. An element f of this type can be combined with an element a of A to give an element $f(b)$ of B. Of course, the elements of this type are thought of (and usually are) functions from A to B.

Now, a type system may or may not have all kinds of constructs. One of these constructs

is *dependent types*. A dependent type is a type which depends on values of other types. For example, $\texttt{array}(T, n)$, the type of arrays of length n, with elements of type T. The study of type systems that have such dependent types is called *dependent type theory*.

Suppose that we have a type A, and a dependent type which we will write $B : A \to \mathbf{Type}$.

One of the possible constructs in a type system is a type $\sum_{a:A} B(a)$ called the *dependent sum*, consisting of all pairs (a, b) with $a : A$ and $b : B(a)$. So every element of $\sum_{a:A} B(a)$ gives an element of one $B(a)$ (for some $a : A$). Note that for the constant dependent type $B(a) = B$, $\sum_{a:A} B$ is the product type $A \times B$.

Another construct which may exist, is a type $\prod_{a:A} B(a)$, consisting of all 'functions' f which map elements $a : A$ to elements $f(a) : B(a)$. Every element of $\prod_{a:A} B(a)$ gives therefore elements of all the $B(a)$ simultaneously. Note that $\prod_{a:A} B$ is the function type $A \to B$.

Lastly, there is a very strong connection between logic and type theory. This is called the *Curry-Howard correspondence* or sometimes referred to as *products as types*. We can view a type T as the proposition "T has an element". An element of T is then a 'proof' or 'witness' of this proposition. If the type system is strong enough, it allows us to do mathematics in it, where:

- A function $f : A \to B$ that takes an argument of type A and yields a result of type B corresponds to the proof of B under the hypothesis that A holds: "suppose that A, then B". Note that the notation $A \to B$ also makes sense if we think of A and B propositions.
- The empty type $\mathbf{0}$ corresponds to 'false'. Note that for all types A, we can construct a function $\mathbf{0} \to A$. In other words, we can prove everything from 'false'.
- The unit type $\mathbf{1}$ corresponds to 'true'.
- The negation of T is the function type $T \to \mathbf{0}$.
- The conjuction "A and B" is given as the product type $A \times B$.
- The disjunction "A or B" is given as the propositional truncation of the coproduct, union or sum type $A \sqcup B$.
- The statement "For all $a : A$, $B(a)$ holds", for some dependent type B (i.e. predicate) over A, is given as the dependent product $\prod_{a:A} B(a)$.

If we think of types as propositions, the question whether some axiom is assumed or not becomes the question whether we assume some (family of) type(s) to be inhabited. Note that most of the axioms that we list here require the notion of mere propositions, propositional truncation and mere existence, which we will cover from Section 3.4 onwards.

- If for all mere propositions A, the type $\|A \vee (A \to \mathbf{0})\|$ is inhabited, we say that the type system assumes the *axiom of excluded middle*: "Either A is true, or A is not true". This is axiom allows us to prove A from 'not not A'.
- If for all (homotopy) sets A, dependent sets $B(a)$ and dependent propositions $C(a, b)$, the type
$$\left(\prod_{a:A} \exists(b : B), C(a, b)\right) \to \exists(f : A \to B), \prod_{a:A} C(a, f(a))$$
is inhabited, we say that the type system assumes the *propositional axiom of choice*: "The product of a family of nonempty sets is nonempty".
- In any type theory which has the required constructs, the following holds: For all dependent types C over A and B, the type
$$\left(\prod_{a:A} \sum_{b:B} C(a, b)\right) \to \sum_{f:A\to B} \prod_{a:A} C(a, f(a))$$
is inhabited. This axiom (or actually more of a theorem) is called the *type theoretic axiom of choice*.

3.2 The Univalence Principle

"*Isomorphic objects are equivalent.*"

This principle is visible in most of mathematics: Sets with a bijection have the same number of elements, isomorphic groups have the same properties, and since the universal property of limits makes them "unique up to unique isomorphism", we can talk about 'the' limit of some diagram in a category.

Now, the *univalence principle* takes this a step further. It states that

"*Isomorphic objects are equal.*"

Univalent foundations seeks to be a foundation for mathematics that is in line with this principle. This is often done within the framework of 'Martin-Löf dependent type theory', a type theory developed by Per Martin-Löf [Mar71]. It is a family of dependent type systems with dependent products, dependent sums, an empty type, a unit type and union types. It is a constructive type theory, so it does not automatically assume the axiom of excluded middle or the propositional axiom of choice. It is however compatible with these axioms, so one can still assume these alongside its usual axioms.

It is important to note that Martin-Löf type theory has *identity types*: given a type T and elements $x, y : T$, we have a type $\mathrm{Id}_T(x, y)$ (note that this is a dependent type), which we will usually denote with $x = y$. An element $p : x = y$ is a proof that x is 'equal' to y. This type comes with an interesting induction principle called *path induction*: we can show any statement about paths $p : x = y$ for generic x and y, if we can show it about the 'trivial' path $\mathrm{refl} : x = x$ for generic x. For example, symmetry of the equality boils down to a function $\prod_{xy:T} x = y \to y = x$. We construct this using path induction with the function that sends $\mathrm{refl} : x = x$ to itself. For more information, see [Uni13], Section 1.12.1.

In set-theoretic mathematics, there is the concept of a 'bijection' $S \simeq T$ of sets (or an isomorphism in the category **Set**), which is often treated as an equivalence. It consists of functions $f : S \to T$ and $g : T \to S$ with $f \cdot g = \mathrm{id}_S$ and $g \cdot f = \mathrm{id}_T$. In type theory, we have a similar concept, which is called 'equivalence' (of types) $S \simeq T$. Since bijections are not well-behaved for types that are not sets (see Section 3.4), because in those cases, a function $f : S \to T$ can have multiple distinct inverses. Therefore, we define $S \simeq T := \sum_{f:S\to T} \mathrm{isequiv}(f)$, for some predicate $\mathrm{isequiv} : (S \to T) \to \mathbf{Type}$ (see [Uni13], Equation 2.4.10). However, intuitively we can still think of these as bijections.

Using the identity type, we can make our statement of the univalence principle more precise (and a bit stronger). For types, we can construct a function

$$\mathrm{idtoequiv} : \prod_{S,T:\mathbf{Type}} (S = T) \to (S \simeq T).$$

We construct this function using path induction with the identity bijection $\prod_S \mathrm{id}_S : (S \simeq S)$. In fact, this is a specific case of the following: for a category C, if we denote the type of isomorphisms between objects c and d with $c \cong d$, we can construct a function

$$\mathrm{idtoiso} : \prod_{c,d:C} (c = d) \to (c \cong d),$$

using path induction with the identity isomorphism $\prod_{c:C} \mathrm{id}_c : (c \cong c)$. We can then formulate the univalence principle for categories as

"For all $c, d : C$, $\mathrm{idtoiso}_{c,d} : (c = d) \to (c \cong d)$ is an equivalence."

A category that adheres to the univalence principle is called a *univalent category*.

Note that if B is a univalent category and A is any category, the functor category $[A, B] = B^A$ is univalent as well. In particular, the presheaf category $PA = [A^{\mathrm{op}}, \mathbf{Set}]$ is univalent.

Also, if A is univalent, any full subcategory $B \subseteq A$ is also univalent, because isomorphisms and equalities of objects in B can be shown to be the 'the same' as their isomorphisms and equalities in A.

Lemma 3.1. *For every category C, there exists a univalent category D with a weak equivalence*

$$\iota : C \hookrightarrow D.$$

Proof. One construction takes D to be the full subcategory of the presheaf category PC of objects that are isomorphic (there *merely exists* an isomorphism) to the Yoneda embedding $よ(X)$ of some object $X : C$. See [AKS15], Theorem 8.5. □

We call such a univalent category a *Rezk completion* of C. Actually, we can call it *the* Rezk completion:

Lemma 3.2 (`rezk_completion_unique`)**.** *A Rezk completion is unique.*

Proof. Precomposition with the weak equivalence $\iota : C \hookrightarrow D$ gives an adjoint equivalence on the functor categories $[D, E] \simeq [C, E]$ for any univalent category E. From this, it follows that ι is the "initial functor" from C: any functor from C to a univalent category factors uniquely through ι.

Now, two initial functors $F : C \to D$ and $F' : C \to D'$ factor uniquely through each other:

$$F' = F \bullet G \quad \text{and} \quad F = F' \bullet G'$$

for two functors G and G' and since F and F' also factor uniquely through themselves, we have

$$G \bullet G' = \mathrm{id}_D \quad \text{and} \quad G' \bullet G = \mathrm{id}_{D'}.$$

From this, we can construct an equivalence $D \cong D'$. Since D and D' are univalent, this equivalence gives an equality of categories, and because $F \bullet G = F'$, we have an equality of pairs

$$(D, F) = (D', F').$$

□

3.3 The Univalence Axiom

Now, even for a basic category, like the category of types, it seems impossible to prove that the univalence principle holds. However, this is no surprise: It turns out that there is a model in set theory for Martin-Löf type theory with univalence [KL18], but also for Martin-Löf type theory with proof irrelevance [MW03]. In the latter, the type $x = y$ has at most one element for all $x, y : T$ and all types T, which contradicts univalence. Therefore, univalence is independent of the axioms of Martin-Löf type theory: both univalence or its negation cannot be proved, but either can be assumed as an axiom.

As mentioned before, univalent foundations attempts to develop as much of mathematics as possible along the univalence principle. Therefore, we assume as our first axiom the *univalence axiom*:

Axiom. *For all S, T :* **Type**, *the function* $\mathrm{idtoequiv}_{S,T} : (S = T) \to (S \simeq T)$ *is an equivalence.*

In other words:

Axiom. Type *is univalent.*

Remark 3.3. One consequence of the independence of the univalence axiom is that equivalent objects are 'indiscernible'. That is: even if we do not yet assume the univalence axiom, we cannot formulate a property that is satisfied by some type, but not by another, equivalent type. This is because such a property would yield a contradiction when we would assume the univalence axiom.

Now, the question arises: how about the univalence axiom for categories other than **Type**? Do we need to keep assuming an additional axiom for every category that we want to be univalent? It turns out that this is not necessary. In practice, most categories consist of 'sets (or types) with additional structure'. For example: topological spaces, groups, λ-theories and algebraic theory algebras. In such categories, we can leverage the univalence of **Type** to show that for isomorphic objects, their underlying types are equal. Also morphisms are usually defined in such a way that they 'preserve' the 'additional structure', which is what we need to show that the category is univalent.

Also, Theorem 4.5 in [AKS15] shows that if a category B is univalent (in the paper, categories are called 'precategories' and univalent categories are just called 'categories'), then the functor category $A \to B$ is also univalent. In particular, the category of (pre)sheaves $A \to \mathbf{Set}$ is univalent.

Therefore, the univalence axiom is a very powerful axiom, and we usually do not need to assume additional axioms to show that more categories satisfy the univalence principle.

The last structure in this section for which we want to consider the univalence axiom, is the 2-category **Cat** of categories. In general, we cannot show that this satisfies the univalence principle. However, we will restrict our attention to the sub-2-category of univalent categories, which are the categories that we want to study. Then Theorem 6.8 in [AKS15] shows that for univalent categories C and D, there is an equivalence between $C = D$ and $C \simeq D$, where $C \simeq D$ denotes the type of (adjoint) equivalences of categories (see Definition 2.4).

Lastly, a result about univalent categories that we will use a couple of times in this thesis:

Lemma 3.4. *For a functor between univalent categories $F : A \to B$, the types 'F is an adjoint equivalence' and 'F is a weak equivalence' are equivalent propositions (see Section 3.4).*

Proof. See [AKS15], Lemma 6.8. □

3.4 Propositions and Sets

If we have a type T and objects x and y, we can wonder how many elements $x = y$ has. In set-based mathematics, this would be a nonsensical question: two elements of a set are either equal or not equal. Therefore, we can expect the answer to be that $x = y$ has at most one element. And indeed, if we do not assume the univalence axiom, we can assume another axiom, called 'uniqueness of identity proofs', which states that for $p, q : x = y$, we have a proof of equality $h : p = q$.

On the other hand, suppose that we do assume the univalence axiom. Consider the two-element type $T = \{-1, 1\}$. We can construct two different equivalences $\mathrm{id}_T, \sigma : T \simeq T$:

$$\mathrm{id}_T(x) = x \quad \text{and} \quad \sigma(x) = -x.$$

By the univalence axiom, we must have that the identity type $(T = T)$ has (at least) two distinct elements, corresponding to id_T and σ. Therefore, the univalence axiom is not compatible with uniqueness of identity proofs, and we see that in a univalent setting, some identity types have more than one element.

A consequence of this is that types in general have too little structure to serve as a foundation for mathematics that was originally set-based. For example, suppose that we want to formalize the theory of groups. A group homomorphism $f : \mathbf{Grp}(H, G)$ is defined as a function on the underlying types $f_1 : H \to G$, together with a proof that it commutes with the group operations: $f_2 : \prod_{x,y:H} f_1(x \circ y) = f_1(x) \circ f_1(y)$. Now, normally in group theory, to show that two homomorphisms $f, g : \mathbf{Grp}(H, G)$ are equal, we show that $\prod_{x:H} f_1(x) = g_1(x)$, the 'data' is equal. However, if we are working with types instead of sets, we also need to show that the proofs of the 'properties' are equal: $f_2 = g_2$ (note that we actually would need

to transport f_2 here, for this equality to typecheck). This makes showing equality of morphisms much more complicated for concrete groups, and sometimes outright impossible for generic groups.

To deal with this, we need the concepts of mere propositions and (homotopy) sets:

Definition 3.5. A *mere proposition* is a type T such that for all $x, y : T$, $x = y$.

Definition 3.6. A *homotopy set* is a type T such that for all $x, y : T$, $x = y$ is a mere proposition.

Since the identity types for a homotopy set are mere propositions, a homotopy set mimics a set in set theory, where equality between elements 'is' or 'is not'. If we restrict the underlying type of a group to be a homotopy set, it can be shown that $\prod_{x,y:H} f_1(x \circ y) = f_1(x) \circ f_1(y)$ is a mere proposition, so $f_2 = g_2$ trivially. This is often true when translating definitions from set theory to univalent foundations: if we base our objects on homotopy sets instead of types, we only have to worry about equality of 'data', the equality of 'properties' follows automatically.

For similar reasons, we restrict the hom-types $C(c, d)$ of categories to be sets. Note that **Type** is not a category under this definition (it is a 'precategory', for some definition of precategory), because in general, the type of functions between sets $S \to T$ is not a set.

3.5 Truncation and (Mere) Existence

As mentioned before, if we want to do mathematics in a dependent type theory, we can 'encode' propositions as types. The elements of the type correspond to the proofs that the proposition is true, see for example the identity types. However, we need to be careful here about the distinction between types in general and mere propositions:

A proof that a type T is nonempty usually consists of giving an element $t : T$. If we encode the statement "T is nonempty" as T, and if T is not a mere proposition, then "T is nonempty" is not a mere proposition, so it has multiple distinct proofs. In some cases, this is exactly what we want, because we want to retrieve the chosen element of T. However, in some cases, we want the fact that a set is nonempty (or any statement in general) to be a mere proposition, to avoid having to carry around a specific element and having to prove equality of two specific elements. For such cases, we have the 'propositional truncation':

Definition 3.7. For a type T, a type $\|T\|$ exists ([Uni13], Section 3.7) with the properties that for all $t : T$, we have an element $|t| : \|T\|$ and that $\|T\|$ is a mere proposition. It has a recursion principle stating that for a mere proposition B, a function $f : T \to B$ induces a function $|f| : \|T\| \to B$ that commutes with $|\cdot|$. This object is called the *propositional truncation*.

The propositional truncation forgets the details about a type, and only keeps the information about whether it is inhabited or not. The recursion principle means that if we are trying to prove a mere proposition based on some element $|t| : \|T\|$, we can pretend that we do have a concrete element $t : T$.

There also is the concept of higher order truncations. For example, the *set truncation* $\|A\|_0$, which is a homotopy set and has an equivalence $(\|A\|_0 \to B) \simeq (A \to B)$ for any set B. However, these higher truncation types become increasingly harder to construct, and in this thesis, we will only need to consider the propositional truncation.

Often, when we prove a theorem or lemma that "there exists some $x : X$ such that $Y(x)$", what we actually mean is that we can construct an element $x : X$ and then an element $y : Y(x)$ of the dependent type Y over X. This is equivalent to constructing an element of $\sum_{x:X} Y(x)$. However, this is in general not a mere proposition. If we want to express that the set of such x is nonempty as a mere proposition, we talk about *mere existence*:

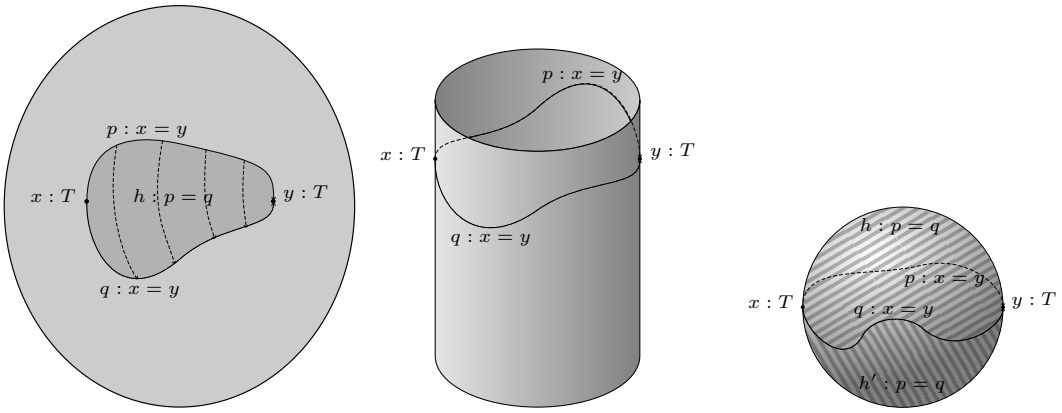


Figure 3.1: Different possible homotopy structures

Definition 3.8. Given a dependent type Y over X, if we say that there *merely exists* an element $x : X$ such that $Y(x)$, we mean that we have an element of the propositional truncation

$$h : (\exists x : Y(x)) := \left\| \sum_{x:X} Y(x) \right\|.$$

For example, if we have objects in a category $c, d : C$ and we want to talk about a retraction f of c onto d, we commonly define this as "a morphism $f_1 : C(c, d)$, such that there exists a 'section': a morphism $f_2 : C(d, c)$ with $f_2 \cdot f_1 = \mathrm{id}_d$". Now, we commonly consider f_1 to be the 'data' of the retraction; we consider retractions f and f' to be the same retraction if $f_1 = f'_1$. This means that being a retraction is about the 'mere existence' of a section. Note, however, that in this case, we cannot use the section in constructions, except when we are trying to prove mere propositions.

Note that for the Curry-Howard correspondence, the product or conjunction $A \wedge B = A \times B$ of two mere propositions is again a mere proposition. However, the union $A \sqcup B$ is not necessarily a mere proposition. To make it into a mere proposition, we need to take the propositional truncation $A \vee B = \|A \sqcup B\|$.

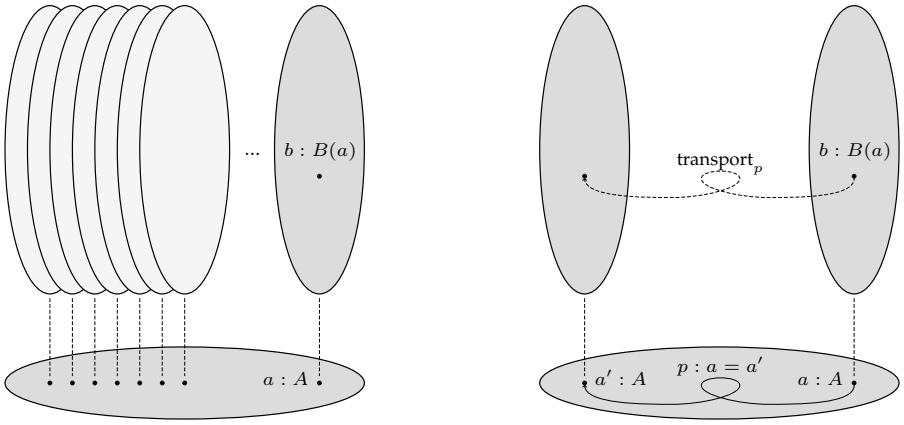
3.6 Equality and Homotopy

Univalent Foundations is often mentioned together with Homotopy Type Theory, because they are related but distinct concepts. Therefore, we will mention it here.

As we saw before, if we do not assume uniqueness of identity proofs, given two elements $x, y : T$, we can have multiple distinct proofs that $x = y$, which is counterintuitive. One way to think about this, is the perspective of homotopy type theory. In homotopy type theory, one considers a type T to be a 'space', intuitively similar to a topological space, but without an explicit topology. The elements $t : T$ are then the points of the space. Elements of the identity type $(s = t)$ are interpreted as 'paths' from s to t. That is why the induction principle of $(s = t)$ is called 'path induction'. Of course, we can go higher: for $p, q : x = y$, the elements $(p = q)$ are 'paths between paths', (path) 'homotopies', 'sheets' or '1-cells', for homotopies $h, h' : p = q$, the elements of $h = h'$ are paths between paths between paths, 'volumes' or '2-cells' etc.

If we have a 'geometric' interpretation of our type theory, we can investigate the 'shape' of a (nonempty) type T, given by the structure of the (higher) identity types.

First of all, if we have $x, y : T$ for which $x = y$ does not have an inhabitant, x and y lie in different 'connected components'. Now, we focus on a connected type:



(a) (b)

Figure 3.2: A fibration with a path in the base space, giving rise to transport in the fibration

For example, are all elements $x, y : T$ of the type equal to each other, and are all elements $p, q : x = y$ of all the (higher) identity types also equal in a unique way? Then we have a *contractible type*, which intuitively looks somewhat like a plane (Figure 3.1a).

It is also possible that any two elements $x, y : T$ are equal, but that there are distinct paths $p, q : x = y$, with no homotopy between them. Then intuitively T looks like a circle or a tube or a projective space, or something more complicated (Figure 3.1b).

Remark 3.9. Note that we can give the type of paths $x = x$ a group structure and $x = y$ is a 'torsor' for this group. If $x = y$ has exactly two distinct elements, the group $x = x$ looks like $\mathbb{Z}/2\mathbb{Z}$, which suggests some projective plane-like structure.

For something to look like a circle or tube, we need this group $x = x$ to be isomorphic to $\mathbb{Z}$. For an example, see [Bez+20].

As a third example, consider a type T in which any two elements $x, y : T$ are equal, and any two paths $p, q : x = y$ have two distinct homotopies $h, h' : p = q$ between them. Then we can imagine the type to look somewhat like a sphere (Figure 3.1c) or something more complicated.

This is the lens through which homotopy type theory studies types.

3.7 Transport

Suppose that we have a dependent type $B : A \to UU$. Intuitively, B is a collection of spaces, lying over the points of A (Figure 3.2a). Now, if we have a path $p : a = a'$ in A, we can use path induction to get a function transport$_p : B(a) \to B(a')$ (Figure 3.2b).

For example, this allows us to transfer an element from $\mathtt{array}(T, n + n)$ to $\mathtt{array}(T, 2 \cdot n)$ with transport over the path $n + n = 2 \cdot n$. Also, if we assume the univalence axiom, and we have, for example, an isomorphism of groups $f : G \cong G'$ and a proof h that G is semisimple, we can transport h along the equality given by f to a proof that G' is semisimple.

Example 3.10. In addition, consider the following example. We take A to be a type universe, and let $B : T \mapsto \mathtt{array}(T, 3)$. Consider the types $a = \{\top, \bot\}$ and $a' = \{0, 1\}$. We have equivalences $\overline{p}$ and $\overline{q}$ between a and a':

$$\overline{p}(\top) = 1, \overline{p}(\bot) = 0 \quad \text{and} \quad \overline{q}(\top) = 0, \overline{q}(\bot) = 1.$$

That means that we have distinct equalities $p, q : a = a'$. Now, suppose that we have an array $x = [\top, \top, \bot] : B(a)$. Since $a = a'$, we would want to treat x as an element of $B(a')$, but then we need to make a choice whether we treat it as $[1, 1, 0]$ or $[0, 0, 1]$ (or something else

altogether). This is exactly the question whether we transport along p or q, and therefore, when our base type is not a set, it is important to be aware that we are transporting a property along an equality, and we need to be careful which equality it is that we transport along.

Another place where transports occur frequently is when proving equality $(x, y) = (x', y')$ of elements of a dependent sum $\sum_{a:A} B(a)$. We can start componentwise by proving $p : x = x'$, but after this, we cannot directly prove $q : y = y'$, because these two live in different types: $B(x)$ and $B(x')$ respectively. Therefore, we need to transport, and then prove

$$\mathsf{transport}_p(y) = y' : B(x').$$

3.7.1 Caution: 'transport hell'

Now, it seems that we can transport all properties and data along equalities. And of course, that is true, but some caution needs to be taken with this when working with a proof assistant.

For example, consider the situation in Example 3.10. As mentioned, we can transport x to get an element $y := \mathsf{transport}_p(x)$, which is an element of $B(a')$, but now suppose that we want to compute something using its first coefficient y_1. Of course, on paper it quickly becomes clear that $y_1 = 1$, but in practice, it takes quite some work to convince a proof assistant of that fact. Of course, we could write a lemma which states that for any array x, any equivalence $\overline{p}$ with corresponding equality p, and any index i,

$$\mathsf{transport}_p(x)_i = \overline{p}(x_i).$$

However, at that point, we have more or less constructed our own function between $B(a)$ and $B(a')$, which is much easier to work with than $\mathsf{transport}_p$.

Experience teaches that in general, it is fine to transport properties $b : B(a)$ of which we will never need the value, only the fact that it (merely) exists. On the other hand, for properties and constructions of which we might later want to use the actual value, it is much better to transfer them 'by hand'. Often, but not always, this criterion coincides with $B(a)$ being a mere proposition.

Another case where unwanted transports often occur is when showing the equality of two complicated elements of a type. For example, $((x_1, x_2), (x_3, x_4)) = ((x'_1, x'_2), (x'_3, x'_4))$, both elements of

$$\sum_{(x_1,x_2):\sum_{a:A} B(a)} \sum_{x_3:C(x_1)} D(x_1, x_2, x_3).$$

This example involves proofs:

- $p : x_1 = x'_1$;
- $q : \mathsf{transport}_p(x_2) = x'_2$;
- $r : \mathsf{transport}_{(p,q)}(x_3) = x'_3$, which can be simplified to $\mathsf{transportf}_p(x_3) = x'_3$, since x_3 does not depend on x_2
- and finally $s : \mathsf{transport}_{((p,q),r)}(x_4) = x'_4$.

In general, these latter proofs r and s are very challenging and best avoided whenever possible. The situation in which one has to work with such complicated transports, which make a proof (seemingly unnecesarily) complicated is called *transport hell*. Some mathematical tools have been developed (see for example Section 7.4 about displayed categories) that help avoid transport hell in some cases. Additionally, the structure or 'strategy' of a proof can sometimes be changed in order to avoid the worst of the transports. However, it is often not possible to avoid transports altogether, and this is one of the reasons why proving something in a proof assistant takes a lot more work than proving it on paper.

Chapter 4

Algebraic Structures

4.1 Algebraic Theories

Definition 4.1 (`algebraic_theory`)**.** We define an *algebraic theory* T to be a sequence of sets T_n indexed over $\mathbb{N}$ with for all $1 \leqslant i \leqslant n$ elements ("variables" or "projections") $x_{n,i} : T_n$ (we usually leave n implicit), together with a substitution operation

$$\_ \bullet \_ : T_m \times T_n^m \to T_n$$

for all m, n, such that

$$\begin{aligned} x_j \bullet g &= g_j \\ f \bullet (x_{l,i})_i &= f \\ (f \bullet g) \bullet h &= f \bullet (g_i \bullet h)_i \end{aligned}$$

for all $1 \leqslant j \leqslant l$, $f : T_l$, $g : T_m^l$ and $h : T_n^m$.

Remark 4.2. For equivalent definitions of different kinds, see Chapter A.

Definition 4.3 (`algebraic_theory_morphism`)**.** A *morphism* f between algebraic theories T and T' is a sequence of functions $f_n : T_n \to T'_n$ (we usually leave the n implicit) such that

$$\begin{aligned} f_n(x_j) &= x_j \\ f_n(f \bullet g) &= f_m(f) \bullet (f_n(g_i))_i \end{aligned}$$

for all $1 \leqslant j \leqslant n$, $f : T_m$ and $g : T_n^m$.

Together, these form the category of algebraic theories **AlgTh** (`algebraic_theory_cat`).

Lemma 4.4 (`limits_algebraic_theory_cat`)**.** *We can construct binary products of algebraic theories, with sets* $(T \times T')_n = T_n \times T'_n$*, variables* (x_i, x_i) *and substitution*

$$(f, f') \bullet (g, g') = (f \bullet g, f' \bullet g').$$

In the same way, the category of algebraic theories has all limits.

Note that the forgetful functor $F :$ **AlgTh** $\to$ **Set**$^{\mathbb{N}}$ *creates limits, since any diagram* d *has a limit and the underlying set of the limit of* d *is the limit of the underlying sets of the objects in* d*.*

Lemma 4.5 (`is_univalent_algebraic_theory_cat`)**.** *Since an isomorphism of algebraic theories* $S \simeq T$ *consists of pointwise isomorphisms* $f_n : S_n \simeq T_n$ *that respect the variables and substitution, the category of algebraic theories is univalent.*

Later on, we will see an example of a trivial algebraic theory (the terminal theory) T, in which every T_n only contains one element. Now, there are many different nontrivial algebraic theories, and it is not easy to find properties that every algebraic theory must satisfy. However, we can show that their variables are distinct, and so every T_n has at least n distinct elements, so no nontrivial algebraic theory is 'almost trivial', if almost trivial means having some N such that every T_n has at most N elements.

Lemma 4.6. *Let T be an algebraic theory, such that T_n has at least two distinct elements for some n. Then for all $1 \leqslant i, j \leqslant m$ with $i \neq j$, we have $x_{m,i} \neq x_{m,j}$.*

Proof. We can also formulate the statement as: If there exist $1 \leqslant i, j \leqslant m$ with $i \neq j$ such that $x_{m,i} = x_{m,j}$, then all T_n contain at most one distinct element.

So, suppose that $x_{m,i} = x_{m,j}$ for some $i \neq j$. For $a, b : L_n$, we define $v : L_n^m$ as

$$v_k = \begin{cases} a & k = i \\ b & k \neq i \end{cases},$$

so in particular, $v_j = b$. Then we have

$$a = v_i = x_{m,i} \bullet v = x_{m,j} \bullet v = v_j = b,$$

so every T_n contains exactly one element, and T is trivial. □

4.2 Algebras

Definition 4.7 (`algebra`)**.** An *algebra* A for an algebraic theory T is a set A, together with an action

$$\bullet : T_n \times A^n \to A$$

for all n, such that

$$x_j \bullet a = a_j$$
$$(f \bullet g) \bullet a = f \bullet (g_i \bullet a)_i$$

for all j, $f : T_m$, $g : T_n^m$ and $a : A^n$.

Definition 4.8 (`algebra_morphism`)**.** For an algebraic theory T, a *morphism* f between T-algebras A and A' is a function $f : A \to A$ such that

$$f(t \bullet a) = f \bullet (f(a_i))_i$$

for all $t : T_n$ and $a : A^n$.

Together, these form the category of T-algebras $\mathbf{Alg}_T$ (`algebra_cat`).

Remark 4.9. The category of algebras has all limits. The set of a limit of algebras is the limit of the underlying sets, so the forgetful functor to the category of sets creates limits.

Lemma 4.10 (`is_univalent_algebra_cat`)**.** *Note that just like with algebraic theories, the category of T-algebras is univalent because its (iso)morphisms preserve $\bullet$.*

Remark 4.11. The notions of algebraic theories and their algebras stem from the field of *universal algebra*. In universal algebra, one studies the 'collections of algebraic structures'. For example, the collection of monoids: a monoid is a set with an associative binary operation and an identity element. Other examples of collections are those of abelian monoids, (abelian) groups or (commutative) rings. As we will see in Example 4.46, one can construct an algebraic theory T such that the category of monoids is equivalent to $\mathbf{Alg}_T$, and we can do very

similar things for the other structures mentioned. Therefore, one could say that universal algebra studies algebraic theories and their algebras (or 'models').

Any category that is equivalent to $\mathbf{Alg}_T$ for some algebraic theory T is called *algebraic*. For example, in Remark 4.38, we will see that **Set** is algebraic. By the remark above, an algebraic category has all limits. It turns out that it also has all colimits (see Lemma 6.51 for binary coproducts, or [ARV10], Part 1, Theorem 4.5 for general colimits). Note, however, that the proof for colimits is a lot more complicated than the proof for limits, just like colimits of algebraic objects usually are more complicated than limits. For example, consider (binary) products and coproducts of groups.

Definition 4.12 (`algebra_pullback`)**.** If we have a morphism of algebraic theories $f : T' \to T$, we have a *pullback functor of algebras* $f^* : \mathbf{Alg}_T \to \mathbf{Alg}_{T'}$. It endows T-algebras with an action from T' given by $g \bullet_{T'} a = f(g) \bullet_T a$. Then T-algebra morphisms commute with this T'-action, so we indeed have a functor.

Remark 4.13. Note that by Lemma A.7, algebras for T are equivalent to finite-product-preserving functors from its Lawvere theory to **Set**. Then $f : T' \to T$ corresponds to a functor on the Lawvere theories $\mathbf{L}_f : \mathbf{L}_{T'} \to \mathbf{L}_T$, and $f^* : \mathbf{Alg}_T \to \mathbf{Alg}_{T'}$ corresponds to precomposition with $\mathbf{L}_f$:

$$\begin{array}{ccc} \mathbf{L}_{T'} & \xrightarrow{\mathbf{L}_f} & \mathbf{L}_T \\ & \searrow & \downarrow \\ & & \mathbf{Set} \end{array}$$

Actually, we can recover some information about the algebraic theory morphism from this pullback functor. For example,

Lemma 4.14. *If f^* is an equivalence of categories, f is an isomorphism.*

Proof. This proof uses the theory algebra, which will be properly defined in Example 4.53.

Note that to show that f is an isomorphism, we only need to show that the f_n are bijections. From the fact that we have inverse functions $g_n : \mathbf{Set}(T, T')$ for the f_n, and the fact that f is an algebraic theory morphism, we know that $(g_n)_n$ is an algebraic theory morphism as well, and that it is the inverse of f.

First of all, note that for all $n : \mathbb{N}$ and $B : \mathbf{Alg}_{T'}$, the following diagram in **Set** commutes:

$$\begin{array}{ccc} \mathbf{Alg}_T(T_n, B) & \xrightarrow{f^*} & \mathbf{Alg}_{T'}(f^*T_n, f^*B) \\ \downarrow{\scriptstyle \sim} & & \downarrow{\scriptstyle \bar{f}_n \cdot -} \\ B^n & \xrightarrow{\sim} & \mathbf{Alg}_{T'}(T'_n, f^*B) \end{array}$$

with $\bar{f}_n : T'_n \to f^*T_n$ the T'-algebra morphism with underlying function f_n. Note that f^* is fully faithful, so the function at the top is a bijection, and so precomposition by $\bar{f}_n$ is a bijection as well. Since f^* is essentially surjective, we have for all $A : \mathbf{Alg}'_T$, some $B : \mathbf{Alg}_T$ and an isomorphism $h : f^*B \xrightarrow{\sim} A$, which gives the following commutative diagram:

$$\begin{array}{ccc} \mathbf{Alg}_{T'}(f^*T_n, f^*B) & \xrightarrow[-\cdot h]{\sim} & \mathbf{Alg}_{T'}(f^*T_n, A) \\ {\scriptstyle \sim}\downarrow{\scriptstyle \bar{f}_n \cdot -} & & \downarrow{\scriptstyle \bar{f}_n \cdot -} \\ \mathbf{Alg}_{T'}(T'_n, f^*B) & \xrightarrow[-\cdot h]{\sim} & \mathbf{Alg}_{T'}(T'_n, A) \end{array}$$

so the arrow on the right is an equivalence as well. Taking $A = T'_n$, there exists some $g : \mathbf{Alg}_{T'}(f^*T_n, T'_n)$ such that $\bar{f}_n \cdot g = \mathrm{id}_{T'_n}$. Also, note that

$$\bar{f}_n \cdot g \cdot \bar{f}_n = \bar{f}_n \cdot \mathrm{id}_{f^*T_n}$$

so taking $A = f^*T_n$, we see that

$$g \cdot \bar{f}_n = \mathrm{id}_{f^*T_n},$$

so f_n, the underlying function of $\bar{f}_n$, is a bijection, with inverse g. □

Remark 4.15. Hyland's proof of this fact is almost the same, but in the last part, he uses some category theory. Instead of explicitly constructing the inverse, he notices that the bijection

$$\mathbf{Alg}_{T'}(f^*T_n, A) \xrightarrow[\bar{f}_n \cdot -]{\sim} \mathbf{Alg}_{T'}(T'_n, A)$$

is in fact the image of $\bar{f}_n^{\mathrm{op}}$ under the Yoneda embedding of $\mathbf{Alg}_{T'}^{\mathrm{op}}$ (or of $\bar{f}_n$ under the covariant Yoneda embedding of $\mathbf{Alg}_{T'}$) into the functor category $\mathbf{Alg}_{T'} \to \mathbf{Set}$:

$$ﾖ(\bar{f}_n) : ﾖ(f^*T_n) \to ﾖ(T'_n).$$

Since the Yoneda embedding is fully faithful, this shows that $\bar{f}_n$ is an isomorphism of T'-algebras, so in particular, it is a bijection.

Note that besides the categories $\mathbf{Alg}_T$, we can also consider the category of 'all' algebraic theory algebras together (`algebra_full_cat`). That is, the category C with $C_0 = \sum_{T:\mathbf{AlgTh}} \mathbf{Alg}_T$ and $C((T, A), (T', A'))$ consisting of pairs $(f, f') : \mathbf{AlgTh}(T, T') \times \mathbf{Set}(A, A')$ such that for all $t : T_n$ and $a : A^n$,

$$f'(t \bullet a) = f(t) \bullet (f'(a_i))_i.$$

We then have a functor $P : C \to \mathbf{AlgTh}$, projecting onto the first coordinate.

Lemma 4.16 (`algebra_fibration`)**.** *P is a fibration.*

Proof. Given an algebraic theory morphism $f : \mathbf{AlgTh}(S, T)$ and a T-algebra A_T, Definition 4.12 gives an S-algebra A_S with underlying set A_T. The cartesian morphism is $(f, \mathrm{id}_{A_T}) : C((S, A_S), (T, A_T))$.

It is cartesian because for $(R, A_R) : C$ and $(g, g') : C((R, A_R), (T, A_T))$ and $h : \mathbf{AlgTh}(R, S)$ with $h \cdot f = g$, the required morphism $C((R, A_R), (S, A_S))$ is given by $g' : \mathbf{Set}(A_R, A_S)$. □

4.3 Presheaves

Definition 4.17 (`presheaf`)**.** A *presheaf* P for an algebraic theory T is a sequence of sets P_n indexed over $\mathbb{N}$, together with an action

$$\bullet : P_m \times T_n^m \to P_n$$

for all m, n, such that

$$t \bullet (x_{l,i})_i = t$$
$$(t \bullet f) \bullet g = t \bullet (f_i \bullet g)_i$$

for all $t : P_l$, $f : T_m^l$ and $g : T_n^m$.

Definition 4.18 (`presheaf_morphism`)**.** For an algebraic theory T, a *morphism* f between T-presheaves P and P' is a sequence of functions $f_n : P_n \to P'_n$ such that

$$f_n(t \bullet f) = f_m(t) \bullet f$$

for all $t : P_m$ and $f : T_n^m$.

Together, these form the category of T-presheaves $\mathbf{Pshf}_T$ (`presheaf_cat`).

Lemma 4.19 (`limits_presheaf_cat`)**.** *The category of presheaves has all limits. The nth set $\overline{P}_n$ of a limit $\overline{P}$ of presheaves P_i is the limit of the nth sets $P_{i,n}$ of the presheaves in the limit diagram. So just like with algebraic theories and algebras, the forgetful functor from the category of presheaves to the category of indexed sets creates limits.*

Lemma 4.20 (`is_univalent_algebra_cat`)**.** *Note that just like with algebraic theories and algebras, the category of T-presheaves is univalent because its (iso)morphisms preserve $\bullet$.*

An analogue to Lemma 4.16 shows that, like with algebras, the total category of presheaves is fibered over the category of algebraic theories (`presheaf_fibration`).

4.4 λ-theories

Let $\iota_{m,n} : T_m \to T_{m+n}$ be the function that sends f to $f \bullet (x_{m+n,1}, \ldots, x_{m+n,m})$. Note that

$$\iota_{m,n}(f) \bullet g = f \bullet (g_i)_{i \leqslant m} \quad \text{and} \quad \iota_{m,n}(f \bullet g) = f \bullet (\iota_{m,n}(g_i))_i.$$

For tuples $x : X^m$ and $y : X^n$, let $x + y$ denote the tuple $(x_1, \ldots, x_m, y_1, \ldots, y_n) : X^{m+n}$.

Definition 4.21 (`lambda_theory`)**.** A *λ-theory* is an algebraic theory L, together with sequences of functions $\lambda_n : L_{n+1} \to L_n$ and $\rho_n : L_n \to L_{n+1}$, such that

$$\lambda_m(f) \bullet h = \lambda_n(f \bullet ((\iota_{n,1}(h_i))_i + (x_{n+1})))$$
$$\rho_n(g \bullet h) = \rho_m(g) \bullet ((\iota_{n,1}(h_i))_i + (x_{n+1}))$$

for all $f : L_{m+1}$, $g : L_m$ and $h : L_n^m$.

Remark 4.22. Hyland claims that 'a λ-theory is an algebraic theory L equipped with semi-closed structure'. By 'semi-closed structure', he probably means the structure of a semi cartesian closed category on the Lawvere theory associated to L (Lemma A.6). Because a Lawvere theory has finite products, we even would have a weak cartesian closed category. For more information about weak cartesian closed categories, see Appendix B.

In the appendix, we see that a λ-theory structure with β-equality on an algebraic theory L gives its associated Lawvere theory C a weak cartesian closed structure with the exponential object 1^1 equal to 1. Conversely, we can give L a λ-theory structure with β-equality from a weak cartesian closed structure on C where 1 is a reflexive object. Note that we really need 1 to be a reflexive object to make the construction work.

However, note that a weak cartesian closed structure on C with 1 a reflexive object contains more information than a λ-theory structure with β-equality on L: If we have a λ-theory structure with β-equality on L, then construct from this a weak cartesian closed category structure with $1^1 = 1$ on C and derive from this again a λ-theory structure on L, we get the same λ-theory structure that we started out with. If, on the other hand, we go from a weak cartesian closed structure where 1 is a reflexive object to a λ-theory structure with β-equality and then to a weak cartesian closed structure again, we might end up with a different structure than the one that we started out with. This is because a weak cartesian closed structure is not necessarily unique up to isomorphism. In particular, we construct semi-exponential objects m^n which are equal to m for all $m, n : C$, and derive their data (ev and cur) from just the data of 1^1 in the original weak cartesian closed structure. Therefore, there are enough ways in which the resulting weak cartesian closed structure can differ from the original one.

Note that here, we really need a choice for the exponential objects. It is not enough to just ask for the mere existence of semi-exponential objects. This is because we need the information contained in $\mathrm{ev} : L_2$ and $\mathrm{cur} : L_{n+1} \to L_n$ to define λ and ρ.

Lastly, if L has both β- and η-equality, the weak cartesian closed structure becomes a cartesian closed structure, which is unique, so in that case, giving an algebraic theory L a λ-theory structure with β- and η-equality is the same as giving its associated Lawvere theory a cartesian closed category structure.

Definition 4.23 (`has_β has_η`)**.** We say that a λ-theory L satisfies β-equality (or that it is a λ-theory with β) if $\rho_n \circ \lambda_n = \mathrm{id}_{L_n}$ for all n. We say that is satisfies η-equality if $\lambda_n \circ \rho_n = \mathrm{id}_{L_{n+1}}$ for all n.

Definition 4.24 (`lambda_theory_morphism`)**.** *morphism* f between λ-theories L and L' is an algebraic theory morphism f such that

$$f_n(\lambda_n(s)) = \lambda_n(f_{n+1}(s))$$
$$\rho_n(f_n(t)) = f_{n+1}(\rho_n(t))$$

for all $s : L_{n+1}$ and $t : L_n$.

Together, these form the category of λ-theories **LamTh** (`lambda_theory_cat`).

Lemma 4.25 (`limits_lambda_theory_cat`)**.** *The category of lambda theories has all limits, with the underlying algebraic theory of a limit being the limit of the underlying algebraic theories. Therefore, the forgetful functor to the category of algebraic theories creates limits.*

Lemma 4.26 (`is_univalent_lambda_theory_cat`)**.** *Note that just like with algebraic theories, the category of λ-theories is univalent because its (iso)morphisms preserve ρ and λ.*

Definition 4.27. A λ-theory algebra or presheaf is an algebra or presheaf for the underlying algebraic theory.

4.4.1 The λ-calculus operations

Now, for a λ-theory L, we have variables $x_{n,i} : L_n$ and λ-abstraction $f \mapsto \lambda(f)$. We will sometimes denote $\lambda(f)$ as $\lambda x_{n+1}, f$ for $f : L_{n+1}$. Now, consider the element $\rho(x_{1,1}) : L_2$.

Definition 4.28 (`app'`)**.** Using the substitution, we have binary operations on the L_n, sending $(f, g) : L_n \times L_n$ to $\rho(x_{1,1}) \bullet (f, g) : L_n$. We will denote $\rho(x_{1,1}) \bullet (f, g)$ as fg, and this gives us our application operation.

This means that we can interpret all three operations of the λ-calculus in L.

Lemma 4.29 (`beta_equality`)**.** *The definitions for β- and η-equality in Definition 4.23 correspond to the usual notions.*

Proof. If L has β-equality, we have

$$\lambda(f)g = \rho(\lambda(f)) \bullet (x_1, \ldots, x_n, g) = f \bullet (x_1, \ldots, x_n, g),$$

so we have the usual β-equality. In the same way η-equality of L gives

$$\lambda(\iota_{n,1}(f)x_{n+1}) = \lambda(\rho(f)) = f.$$

□

Remark 4.30 (`app_from_app'`). Also note that for $f : L_n$,

$$\rho(f) = \rho(x_1 \bullet f) = \iota_{n,1}(f)x_{n+1}.$$

Therefore, we can also think of the application as being the primary operation, from which we derive ρ. In the same way, we have

$$\rho^m(f) = \iota_{n,m}(f)x_{n+1} \ldots x_{n+m}.$$

Remark 4.31 (app). Note that for $f, g : L_n$,

$$\rho(f) \bullet (x_1, \ldots, x_n, g) = \rho(x_1) \bullet (\iota_{n,1}(f), x_{n+1}) \bullet (x_1, \ldots, x_n, g) = \rho(x_1) \bullet (\iota_{n,1}(f), g),$$

so we could also define the application as $fg = \rho(f) \bullet (x_1, \ldots, x_n, g)$, although that is more complicated.

Remark 4.32. Now, there are two ways to study λ-calculus-like structures using the tools given here. We can study λ-theories, which by the reasoning above have a λ-calculus structure. Alternatively, we can try to study the λ-calculus-like structures as algebras for some algebraic theory Λ. However, it is not clear beforehand that this will succeed, because there is no reason why this category should be algebraic. In particular, λ-abstraction causes problems, because it is not an algebraic operation. Of course, we can still do exactly the same as in Example 4.46: We can let Λ_n be the set of Λ-terms with constants $x_1, \ldots, x_n$, which we can also regard as free variables (see Definition 4.42), and study the category $\mathbf{Alg}_\Lambda$.

Later on, we will see that these two ways of studying λ-calculus-like structures do coincide: we will see that the category **LamTh** is indeed algebraic, and moreover, that the objects of $\mathbf{Alg}_\Lambda$ are equivalent to λ-theories.

4.5 Examples

There is a lot of different examples of algebraic theories and their algebras. Some of these even turn out to be λ-theories. In this section, we will discuss a couple of these.

4.5.1 The free algebraic theory on a set

Example 4.33 (free_functor). Let S be a set. We can construct an algebraic theory $F(S)$ by taking $F(S)_n = S \sqcup \{1, \ldots, n\}$ with projections $x_i = i$ and substitution

$$i \bullet g = g_i \qquad\qquad s \bullet g = s$$

for $i : \{1, \ldots, n\}$ and $s : S$.

If we have a function $f : S \to S'$, we get a morphism $F(f) : F(S) \to F(S')$ given by

$$F(f)_n(i) = i \qquad\qquad F(f)_n(s) = f(s)$$

for $i : \{1, \ldots, n\}$ and $s : S$.

Also, F obviously respects the identity and substitution morphisms, so it is a functor.

Note that we have a forgetful functor $(\cdot)_0$ that sends a morphism of algebraic theories $g : T \to T'$ to the function $f_0 : T_0 \to T'_0$.

Lemma 4.34 (free_functor_is_free)**.** *The algebraic theory $F(S)$ defined above, is the free algebraic theory on the set S.*

Proof. Let T be an algebraic theory. We have an equivalence

$$\mathbf{AlgTh}(F(S), T) \cong \mathbf{Set}(S, T_0),$$

sending $f : \mathbf{AlgTh}(F(S), T)$ to $f_0 : S = S \sqcup \varnothing \to T_0$ (this is trivially natural in S and T) and $f : \mathbf{Set}(S, T_0)$ to the functions $g_n : F(S)_n \to T_n$ given by

$$g_n(i) = x_i \qquad\qquad g_n(s) = f(s) \bullet ().$$

□

The proofs that $F(S)$ is an algebraic theory and that $F(f)$ and g are algebraic theory morphisms is an easy exercise in case distinction.

Corollary 4.35 (`projections_theory`)**.** *$F(\varnothing)$ is the initial algebraic theory.*

Proof. For $S = \varnothing$, the equivalence of hom-sets becomes

$$\mathbf{AlgTh}(F(\varnothing), T) \cong \mathbf{Set}(\varnothing, T_0)$$

and the latter has exactly one element. □

Lemma 4.36 (`algebra_coslice_equivalence`)**.** *There is an adjoint equivalence between the category $\mathbf{Alg}_{F(S)}$ and the coslice category $S \downarrow \mathbf{Set}$.*

Proof. For the equivalence, we send a $F(S)$-algebra A to the set A with morphism $s \mapsto s \bullet ()$. An algebra morphism $f : A \to B$ is sent to the coslice morphism $f : (S \to A) \to (S \to B)$. This constitutes a functor.

Note that the category of $F(S)$-algebras is univalent.

Also, the functor is fully faithful, since one can show that for $F(S)$-algebras, the coslice morphism $\varphi : (f : S \to A) \to (f' : S \to B)$ also has the structure of an algebra morphism $\varphi : A \to B$.

Lastly, the functor is essentially surjective, since we can lift an object $f : S \to X$ to a $F(S)$-algebra X, with action

$$i \bullet x = x_i \quad \text{and} \quad s \bullet x = f(s).$$

Therefore, the functor $\mathbf{Alg}_{F(S)} \to S \downarrow \mathbf{Set}$ is an adjoint equivalence.

The proofs of these facts work by simple case distinction, and by using the properties of the coslice and algebra morphisms. □

Remark 4.37. $F(\varnothing)$ is, in some sense, the smallest nontrivial algebraic theory. Then $F(S)$ is the smallest nontrivial algebraic theory that has the elements of S as constants.

Remark 4.38. Note that the category of $F(\varnothing)$-algebras is equivalent to the coslice-category $(\varnothing \downarrow \mathbf{Set})$, which, since $\varnothing$ is the initial set, is just equivalent to **Set**.

4.5.2 The free λ-theory on a set

In this subsection, we will use the λ-calculus operations defined in Subsection 4.4.1.

Like with the free algebraic theory, we will construct the free λ-theory as the smallest nontrivial λ-theory (which is the λ-calculus) with some additional constants.

Let S be a set. Consider the sequence of inductive types $(\Lambda(S)_n)_n$ with the following constructors:

$$\begin{aligned}
&\mathtt{Var}_n : \{1, \dots, n\} \to \Lambda(S)_n;\\
&\mathtt{App}_n : \Lambda(S)_n \to \Lambda(S)_n \to \Lambda(S)_n;\\
&\mathtt{Abs}_n : \Lambda(S)_{n+1} \to \Lambda(S)_n;\\
&\mathtt{Con}_n : S \to \Lambda(S)_n.
\end{aligned}$$

Define a substitution operator $\bullet : \Lambda(S)_m \times \Lambda(S)_n^m \to \Lambda(S)_n$ by induction on the first argument:

$$\begin{aligned}
\mathtt{Var}_m(i) \bullet g &= g_i;\\
\mathtt{App}_m(a, b) \bullet g &= \mathtt{App}_n(a \bullet g, b \bullet g);\\
\mathtt{Abs}_m(a) \bullet g &= \mathtt{Abs}_n(a \bullet ((g_i \bullet (x_{n+1,j})_j)_i + (x_{n+1})));\\
\mathtt{Con}_m(s) \bullet g &= \mathtt{Con}_n(s).
\end{aligned}$$

And then quotient $\Lambda(S)$ by the relation generated by

$$\mathtt{App}_m(\mathtt{Abs}_m(f), g) \sim f \bullet ((x_{n,i})_i + (g))$$

for all $f : \Lambda(S)_{n+1}$ and $g : \Lambda(S)_n$.

Example 4.39. We can give the sequence of sets $\Lambda(S)$ an algebraic theory structure with variables $x_{m,i} = \mathtt{Var}_m(i)$ and the substitution operator $\bullet$ defined above. We can give $\Lambda(S)$ a λ-theory structure with β-equality by taking

$$\lambda_n(f) = \mathtt{Abs}_n(f) \quad \text{and} \quad \rho_n(f) = \mathtt{App}_{n+1}(f \bullet (\mathtt{Var}_{n+1}(i))_i, \mathtt{Var}_{n+1}(n+1)).$$

Now, given a function $S \to S'$, we define a morphism $\mathbf{LamTh}(\Lambda(S), \Lambda(S'))$ by induction, sending $\mathtt{Var}(i)$, $\mathtt{App}(a, b)$ and $\mathtt{Abs}(a)$ in $\Lambda(S)$ to their corresponding elements in $\Lambda(S')$ and sending $\mathtt{Con}(s)$ to $\mathtt{Con}(f(s))$.

Note that, like with the previous example, we have a forgetful functor $\dot{()}_0 : \mathbf{LamTh} \to \mathbf{Set}$.

Lemma 4.40. *$\Lambda(S)$ is the free λ-theory on S.*

Proof. Let L be a λ-theory. We have an equivalence

$$\mathbf{LamTh}(\Lambda(S), L) \cong \mathbf{Set}(S, L_0),$$

sending $f : \mathbf{LamTh}(\Lambda(S), L)$ to $f_0|_S : S \to L_0$ (again, trivially natural in S and L) and conversely, $g : \mathbf{Set}(S, L_0)$ to the inductively defined $f : \mathbf{LamTh}(\Lambda(S), L)$ given by

$$\begin{aligned} f(\mathtt{Var}(i)) &= x_i; \\ f(\mathtt{App}(a, b)) &= f(a)f(b); \\ f(\mathtt{Abs}(a)) &= \lambda x_{n+1}, f(a); \\ f(\mathtt{Con}(s)) &= g(s) \bullet (). \end{aligned}$$

□

Remark 4.41. One can also consider this lemma a proof that we can 'interpret' the λ-calculus with some constants inside any λ-theory L if we give an interpretation of the constants as terms in L_0.

The proofs that $\Lambda(S)$ is indeed a λ-theory and that $\Lambda(f)$ and g are λ-theory morphisms, mainly work by definition of $\bullet$, λ and ρ, by induction on the terms of $\Lambda(S)$ and by invoking the properties of the λ-theory L.

Definition 4.42 (`lambda_calculus_lambda_theory`)**.** We define the 'pure' λ-calculus Λ to be $\Lambda(\varnothing)$.

Corollary 4.43. *Λ is the initial λ-theory.*

About Λ-algebra morphisms

Lemma 4.44. *Let A and B be Λ-algebras and let $f : \mathbf{Set}(A, B)$ be a function that preserves the application and the Λ-definable constants:*

$$f((x_1x_2) \bullet (a, b)) = (x_1x_2) \bullet (f(a), f(b)) \quad \textit{and} \quad f(s \bullet ()) = s \bullet ()$$

for all $a, b : A$ and $s : \Lambda_0$. Then f is a Λ-algebra morphism.

Proof. Note that for $s : \Lambda_{n+1}$ and $a : A^{n+1}$,

$$(x_1 x_2) \bullet (\lambda(s) \bullet (a_i)_i, a_{n+1}) = s \bullet a.$$

By induction, we can express $s \bullet a$ using a combination of $(x_1x_2) \bullet (\cdot, \cdot)$ and $\lambda^n(s)$:

$$\begin{aligned} f(s \bullet a) &= f((x_1x_2) \bullet (\dots((x_1x_2) \bullet (\lambda^n(s) \bullet (), a_1), \dots), a_n)) \\ &= (x_1x_2) \bullet (\dots((x_1x_2) \bullet (f(\lambda^n(s) \bullet ()), f(a_1)), \dots), f(a_n)) \\ &= s \bullet (f(a_i))_i, \end{aligned}$$

so f is a Λ-algebra morphism. □

4.5.3 The free object algebraic theory

Example 4.45 (`free_object_theory`, `free_object_algebra_functor`). Take a category C, with a forgetful functor $G : C \to \mathbf{Set}$ and a free functor $F : \mathbf{Set} \to C$. Let $\eta : \mathrm{id}_{\mathbf{Set}} \Rightarrow F \bullet G$ be the unit of the adjunction and let $\varphi : C(F(c), d) \cong \mathbf{Set}(c, G(d))$ be the natural equivalence of homsets.

We define an algebraic theory T with $T_n = G(F(\{1, \dots, n\}))$, projections $x_{n,i} = \eta_{\{1,\dots,n\}}(i)$. For the substitution, note that we take $t_1, \dots, t_m : T_n$, so we have $t : \{1, \dots, m\} \to G(F(\{1, \dots, n\}))$. We then take

$$s \bullet t = G(\varphi^{-1}(t))(s).$$

Now, given an object $c : C$, we can create a T-algebra $\alpha(c)$, with set $G(c)$ and action

$$s \bullet t = G(\varphi^{-1}(t))(s).$$

Also, given a morphism $f : C(c, d)$. This gives a morphism $G(f) : \alpha(c) \to \alpha(d)$. Therefore, $\alpha : C \to \mathbf{Alg}_T$ is a functor.

The proofs that T is an algebraic theory, that $G(c)$ is an algebra and that $G(f)$ is an algebra morphism mainly rely on the fact that φ is natural.

So we have a functor from C to the category of T-algebras. One can wonder whether there also is a functor the other way, or whether α is even an equivalence. If α is an equivalence, C is trivially an algebraic category, but the question for which algebraic categories α is an equivalence is harder to answer.

Of course, for many common categories in algebra, where an object of C is a set with some operations between its elements, one can carefully choose some elements of T_0, T_1, T_2 etc., which act on an algebra like the specified operations, which turns α into an equivalence.

Example 4.46 (`monoid_algebra_equivalence`). For C the category of monoids, $\alpha : C \to \mathbf{Alg}_T$ is an adjoint equivalence.

Note that T_n is the free monoid on n elements. Its elements can be viewed as strings $(x_1x_5x_3x_{18}\dots x_7)$ with the characters $x_1, \dots, x_n$, with the x_i the generators of the monoid, acting as the projections of the algebraic theory.

Let A be a T-algebra. We can give A a monoid structure by taking, for $a, b : A$,

$$ab = (x_1x_2) \bullet (a, b)$$

and unit element

$$1 = () \bullet ().$$

Then the laws like associativity follow from those laws on the monoid and from the fact that the action on the algebra commutes with the substitution:

$$a(bc) = (x_1(x_2x_3)) \bullet (a, b, c) = ((x_1x_2)x_3) \bullet (a, b, c) = (ab)c.$$

Note that if we take a monoid, turn it into a T-algebra and then into a monoid again, we still have the same underlying set, and it turns out that the monoid operation and unit element are equal to the original monoid operation and unit element. Therefore, α is essentially surjective. It is also fully faithful, since any T-algebra morphism respects the action of T, which makes it into a monoid morphism. Therefore, α is an adjoint equivalence.

Remark 4.47. In the same way, one can characterize groups, rings and R-algebras (for R a ring) as algebras of some algebraic theory. On the other hand, one can not use this method to describe fields as algebras for some theory T, because one would need to describe the inverse $z \mapsto z^{-1}$ operation as $t \bullet (z)$ for some $t : T_1$, with $zz^{-1} = 1$, but since the elements of the algebraic theory act on all (combinations of) elements of the algebra, one would be able to take the inverse $0^{-1} = t \bullet (0)$ with $00^{-1} = 1$, which would make no sense.

Remark 4.48. Another counterexample is the category **Top** of topological spaces. We have a forgetful functor $G : \mathbf{Top} \to \mathbf{Set}$ that just forgets the topology. On the other hand, we have a free functor $F : \mathbf{Set} \to \mathbf{Top}$ which endows a set with the discrete topology. The construction above yields the initial algebraic theory $T_n = \{1, \ldots, n\}$, with an algebra action on every topological space $i \bullet (a_1, \ldots, a_n) = a_i$. Now, note that we can endow the set $\{\top, \bot\}$ with four different, nonisomorphic topologies, which all yield the same T-algebra. In other words: the T-algebra structure does not preserve the topological information. Therefore, the functor $\alpha : \mathbf{Top} \to \mathbf{Alg}_T$ is not an equivalence.

4.5.4 The terminal theory

Example 4.49 (`one_point_theory`). We can create a (somewhat trivial) algebraic theory T by taking $T_n = \{\star\}$, with projections $x_i = \star$ and substitution $\star \bullet \star = \star$. Taking $\lambda(\star) = \star$ and $\rho(\star) = \star$, we give it a λ-theory structure (with β and η-equality). Checking that this is indeed an algebraic theory and even a λ-theory is trivial.

Now, given any other algebraic theory T', there exists a unique function $T'_n \to T_n$ for every n, sending everything to $\star$. These functions actually constitute an algebraic theory morphism $T' \to T$. If T' is a λ-theory, the algebraic theory morphism is actually a λ-theory morphism. Again, checking this is trivial.

Therefore, T is the terminal algebraic theory and λ-theory.

Lemma 4.50 (`one_point_theory_algebra_is_trivial`)**.** $\{\star\}$ *is the only algebra of the terminal theory.*

Proof. Let A be a T-algebra. First of all, we have an element $\star_A = \star_T \bullet_0 ()$. Secondly, for all elements $\star, \star' : A$, we have

$$\star = x_1 \bullet (\star, \star') = \star \bullet (\star, \star') = x_2 \bullet (\star, \star') = \star'.$$

Therefore, $A = \{\star\}$, which allows exactly one possible T-action:

$$\star \bullet (\star, \ldots, \star) = \star.$$

□

4.5.5 The endomorphism theory

Definition 4.51 (`endomorphism_algebraic_theory`)**.** Suppose that we have a category C and an object $X : C$, such that all powers X^n of X are also in C. The *endomorphism theory* $E(X)$ of X is the algebraic theory given by $E(X)_n = C(X^n, X)$ with projections as variables $x_{n,i} : X^n \to X$ and a substitution that sends $f : X^m \to X$ and $g_1, \ldots, g_m : X^n \to X$ to $f \circ \langle g_i \rangle_i : X^n \to X^m \to X$.

Definition 4.52 (endomorphism_lambda_theory)**.** Now, suppose that the exponential object X^X exists, and that we have morphisms back and forth $abs : X^X \to X$ and $app : X \to X^X$. Let, for $Y : C$, φ_Y be the isomorphism $C(X \times Y, X) \xrightarrow{\sim} C(Y, X^X)$. We can give $E(X)$ a λ-theory structure by setting, for $f : E(X)_{n+1}$ and $g : E(X)_n$,

$$\lambda(f) = abs \circ \varphi_{X^n}(f) \qquad \rho(g) = \varphi^{-1}_{X^n}(app \circ g).$$

The proofs that $E(X)$ is an algebraic theory and a λ-theory, use properties of the product, and naturality of the isomorphism φ_Y.

4.5.6 The theory algebra

Example 4.53 (theory_algebra). Let T be an algebraic theory and n a natural number. We can endow the T_n with a T-algebra structure, by taking the substitution operator of T as the T-action. Since this commutes with the substitution operator and the projections, T_n is a T-algebra.

Lemma 4.54 (theory_algebra_free)**.** *T_n is the free T-algebra on n generators.*

Proof. Recall from Lemma A.7 that T-algebras are equivalent to finite-product-preserving **Set**-valued functors on the Lawvere theory **L** associated to T. Now, recall from Definition 2.12 for any category C, we can embed C^{op} inside $[C, \mathbf{Set}]$, the category of **Set**-valued functors on C. This embeds $n : \mathbf{L}$ as the theory algebra T_n. Then the Yoneda Lemma gives a bijection

$$\mathbf{Alg}_T(T_n, A) \cong A^n = \mathbf{Set}(\{1, \ldots, n\}, A)$$

natural in n and A. Explicitly, it sends $f : \mathbf{Alg}_T(T_n, A)$ to $(f(x_i))_i$ and $(a_i)_i : A^n$ to $f \mapsto f \bullet a$. The natural equivalence immediately shows that T_n is the free T-algebra on n elements. □

Using some additional machinery, we can combine this with the algebra pullback functor to get another functor:

Definition 4.55. Take a nonnegative integer n. Take an object $S : \mathbf{AlgTh}$. For every object $T : \mathbf{AlgTh}$, we can take the T-algebra T_n. Then every morphism $f : \mathbf{AlgTh}(S, T)$ gives an S-algebra f^*T_n. In fact, this is a functor from $(S \downarrow \mathbf{AlgTh})$ to $\mathbf{Alg}_S$: We send a morphism $g : (S \downarrow \mathbf{AlgTh})((T, f), (T', f'))$ to

$$g_n : f^*T_n \to (f')^*T'_n.$$

This is an algebra morphism because it commutes with f, f' and the substitution of T and T': For $s : S_m$ and $t : (f^*T_n)^m$, we have

$$g_n(s \bullet_{f^*T_n} t) = g_m(f_m(s)) \bullet_{T'} (g_n(t_i))_i = s \bullet_{(f')^*T'_n} (g_n(t_i))_i.$$

The functor obviously preserves the identity morphisms and composition of morphisms, so it is indeed a functor.

4.5.7 The initial presheaf

Example 4.56. Let T be an algebraic theory. We can construct a T-presheaf P, with $P_n = \varnothing$. Then $\bullet : P_m \times T_n^m \to P_m$ is trivial, and the presheaf laws hold trivially.

Lemma 4.57. *This is indeed the initial presheaf.*

Proof. Let Q be a T-presheaf. For all n, since P_n is empty, there is only one possible function $P_n \to Q_n$. These functions trivially satisfy the presheaf morphism laws, so they constitute the unique presheaf morphism $P \to Q$. □

4.5.8 The theory presheaf

Example 4.58 (`theory_presheaf`). Let T be an algebraic theory. We can endow T with a T-presheaf structure, by taking the substitution operator of T as the action on T. Since this commutes with the substitution operator and the projections, T is a T-presheaf.

Lemma 4.59 (`presheaf_to_L`)**.** *We have natural bijections*

$$\mathbf{Pshf}_T(T^n, Q) \cong Q_n$$

for $Q : \mathbf{Pshf}_T$.

Proof. Recall from Lemma A.8 that T-presheaves are equivalent to presheaves on the Lawvere theory $\mathbf{L}$ associated to T. Now, recall from Definition 2.12 that we can embed any category inside its own category of presheaves. This embeds $n : \mathbf{L}$ as the power $T^n = (T^n_m)_m$ of the theory presheaf. Then the Yoneda Lemma gives a bijection

$$\mathbf{Pshf}_T(T^n, Q) \cong Q_n$$

natural in n and Q. Explicitly, it sends $f : \mathbf{Pshf}_T(T^n, Q)$ to $f_n(x_1, \ldots, x_n)$ and $q : Q_n$ to $(t_i)_i \mapsto q \bullet t$. □

4.5.9 The "+1" presheaf

Example 4.60 (`plus_1_presheaf`). [The '+1' presheaf] Given a T-presheaf Q, we can construct a presheaf $A(Q, l)$ with $A(Q, l)_n = Q_{n+l}$ and, for $q : A(Q, l)_m$ and $f : T^m_n$, action

$$q \bullet_{A(Q,l)} f = q \bullet_Q ((\iota_{n,l}(f_i))_i + (x_{n+i})_i).$$

Lemma 4.61 (`theory_presheaf_exponentiable`)**.** *For all l and T-presheaves Q, $A(Q, l)$ is the exponential object Q^{T^l}.*

Proof. We will show that $A(-, l)$ constitutes a right adjoint to the functor $- \times T^l$. We will do this using universal arrows.

For Q a T-presheaf, take the arrow $\varphi : A(Q, l) \times T^l \to Q$ given by $\varphi(q, t) = q \bullet_Q ((x_{n,i})_i + t)$ for $q : A(Q, l)_n = Q_{n+l}$ and $t : T^l_n$.

Now, given a T-presheaf Q' and a morphism $\psi : Q' \times T^l \to Q$. Define $\tilde{\psi} : Q'_n \to A(Q, l)_n$ by $\tilde{\psi}(q) = \psi(\iota_{n,l}(q), (x_{n+i})_i)$.

Then ψ factors as $\varphi \circ (\tilde{\psi} \times \mathrm{id}_{T^l})$. Also, some equational reasoning shows that $\tilde{\psi}$ is unique, which proves that φ indeed is a universal arrow. □

Chapter 5

Previous Work in Categorical Semantics

5.1 The Correspondence Between Categories and Typed λ-calculi

In [SH80], page 413, Scott and Lambek argue that there is a correspondence between simply typed λ-calculi and cartesian closed categories (categories with products and 'function objects').

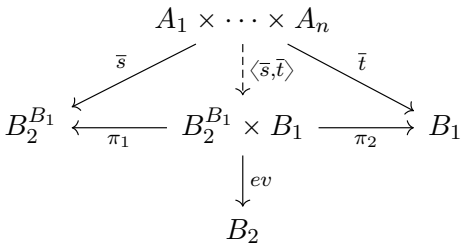
Types in the λ-calculus correspond to objects in the category.

Types $A \to B$ in the λ-calculus correspond to exponential objects B^A in the category.

Terms in the λ-calculus of type B, with free variables $x_1 : A_1, \ldots, x_n : A_n$, correspond to morphisms $A_1 \times \cdots \times A_n \to B$.

A free variable $x_i : A_i$ in a context with free variables $x_1 : A_1, \ldots, x_n : A_n$ corresponds to the projection morphism $\pi_i : A_1 \times \cdots \times A_n \to A_i$.

Given a term $s : B_1 \to B_2$ and a term $t : B_1$, both with free variables $x_1 : A_1, \ldots, x_n : A_n$, corresponding to morphisms $\overline{s} : A_1 \times \cdots \times A_n \to B_2$ and $\overline{t} : A_1 \times \cdots \times A_n \to B_1$, the application $st : B_2$ corresponds to the composite of the product morphism with the evaluation morphism $A_1 \times \cdots \times A_n \to B_2^{B_1} \times B_1 \to B_2$.



Given a term $t : B$ with free variables $x_1 : A_1, \ldots, x_n : A_n$, the abstraction $(\lambda x_n, t) : A_n \to B$ corresponds to using the adjunction corresponding to the exponential object of A_n:

$$C(A_1 \times \cdots \times A_{n-1} \times A_n, B) \simeq C(A_1 \times \cdots \times A_{n-1}, B^{A_n}).$$

5.2 The Category of Retracts

The next sections make extensive use of a category called **R**, which Hyland calls the 'category of retracts'. In this section, we will define the category, and show some properties about it.

Let L be a λ-theory. First of all, for $a_1, a_2 : L_0$, we define

$$\begin{aligned} a_1 \circ a_2 &= \lambda x_1, a_1(a_2 x_1); \\ (a_1, a_2) &= \lambda x_1, x_1 a_1 a_2; \\ \langle a_1, a_2 \rangle &= \lambda x_1, (a_1 x_1, a_2 x_1); \\ \pi_i &= \lambda x_1, x_1(\lambda x_2 x_3, x_{i+1}). \end{aligned}$$

Although, actually, since every one of these starts with a λ-abstraction, we need to lift the constants a_i to $\iota_{0,1}(a_i) : L_1$ to make the definitions above typecheck.

Note that $\pi_i(a_1, a_2) = a_i$ and $\pi_i \circ \langle a_1, a_2 \rangle = \lambda x_1, a_i x_1$, which is exactly what we would expect of a projection.

Also, note that by replacing the x_i by x_{n+i} and the $\iota_{0,1}(a_i)$ by $\iota_{n,1}(a_i)$, we obtain definitions not only for elements of L_0, but for all L_n.

Later on, we will need not only pairs and their projects, but also n-tuples with projections. Therefore, we define

$$\begin{aligned} (a_i)_i = (a_1, \dots, a_n) &= ((\dots((c, a_1), a_2), \dots), a_n); \\ \langle a_i \rangle_i = \langle a_1, \dots, a_n \rangle &= \langle\langle \dots \langle\langle c, a_1 \rangle, a_2 \rangle, \dots \rangle, a_n \rangle; \\ \pi_{n,i} &= \pi_2 \circ \underbrace{\pi_1 \circ \dots \circ \pi_1}_{n-i} \end{aligned}$$

for some constant c, which usually is something like $\lambda x_{n+1}, x_{n+1}$.

Recall from Section 2.11 that we can view any monoid M as a one-object category C_M, where the morphisms are the elements of M.

Definition 5.1 (R)**.** Note that the set of λ-terms without free variables L_0 has a monoid structure under the composition defined above. The category **R** is defined as the Karoubi envelope $\overline{C}_{L_0}$ of the category C_{L_0} of this monoid.

We choose to implement the Karoubi envelope using a slightly different (but equivalent) very syntactical construction using idempotents, because Scott reasons about them in this way:

$$\mathbf{R}_0 = \{A : L_0 \mid A \circ A = A\} \quad \text{and} \quad \mathbf{R}(A, B) = \{f : L_0 \mid B \circ f \circ A = f\},$$

with $\mathrm{id}_A = A$ and composition given by $\circ$.

Remark 5.2 (R_ob_weq_R')*.* Hyland instead defines **R** as the Karoubi envelope $\overline{C}_{L_1}$ (again, the construction using the idempotents) of the monoid $(L_1, - \bullet (-))$ with identity element $x_{1,1}$:

$$\mathbf{R}_0 = \{A : L_1 \mid A \bullet A = A\} \quad \text{and} \quad \mathbf{R}(A, B) = \{f : L_1 \mid B \bullet f \bullet A = f\},$$

writing $s \bullet t$ instead of $s \bullet (t)$ for $s, t : L_1$. Note that we have a monoid morphism:

$$\begin{array}{ccc} L_1 \times L_1 & \xrightarrow{\bullet} & L_1 \\ {\scriptstyle (s,t) \mapsto ((\lambda x_1, s), (\lambda x_1, t))} \downarrow & & \downarrow {\scriptstyle s \mapsto (\lambda x_1, s)} \\ L_0 \times L_0 & \xrightarrow{\circ} & L_0 \end{array}$$

and since for $A : \mathbf{R}$, $A = \lambda x_1, \iota_{0,1}(A)(\iota_{0,1}(A) x_1)$, we have $\lambda x_1, \iota_{0,1}(A) x_1 = A$. This shows that $s \mapsto \lambda x_1, s$ and $t \mapsto \iota_{0,1}(t) x_1$ (both on objects and morphisms) constitute an equivalence between Hyland's category of retracts and Scott's category of retracts.

Now, to give a bit more intuition for the objects of **R**, we can pretend that an object $A : \mathbf{R}$ consists of the set of elements that satisfy $Aa = a$. Then a morphism $f : \mathbf{R}(A, B)$ gives $B(fa) = (B \circ f)a = fa$. This actually constitutes a functor from R to $\mathbf{Pshf}_L$:

Definition 5.3. We define a functor $\varphi : \mathbf{R} \to \mathbf{Pshf}_L$ by taking

$$\varphi(A)_n = \{a : L_n \mid \iota_{0,n}(A)a = a\} \quad \text{and} \quad \varphi(f)_n(a) = \iota_{0,n}(f)a$$

for $A, B : \mathbf{R}$, $f : \mathbf{R}(A, B)$ and $a : A$. The presheaf action on $\varphi(A)$ is given by the substitution of L:

$$(a, f) \mapsto a \bullet f$$

for $f : L_n^m$ and $a : L_m$ such that $\iota_{0,m}(A)a = a$.

It turns out that this is an embedding:

Lemma 5.4. *This functor φ is fully faithful.*

Proof. Take $A, B : \mathbf{R}$. We need to show that $f \mapsto \varphi(f)$ is an equivalence between $\mathbf{R}(A, B)$ and $\mathbf{Pshf}_L(\varphi(A), \varphi(B))$. We have a function

$$\psi : \mathbf{Pshf}_L(\varphi(A), \varphi(B)) \to \mathbf{R}(A, B), \quad g \mapsto (\lambda x_1, g_1(\iota_1(A)x_1))$$

which gives us the inverse. To see that this even typechecks, note that we have

$$\iota_1(A)x_1 : \varphi(A)_1 \quad \text{and} \quad g_1(\iota_1(A)x_1) : \varphi(B)_1 \subseteq L_1.$$

Using the fact that g is a presheaf morphism, we can show that

$$B \circ \psi(g) \circ A = \psi(g),$$

and that φ and ψ are inverses. □

Remark 5.5. Note that for all $A : \mathbf{R}$, we can 'reduce' any $x : L_n$ to

$$\iota_{0,n}(A)x : \varphi(A)_n$$

and in the same way, for all $A, B : \mathbf{R}$, we can turn any $f : L_0$ into a morphism $B \circ f \circ A : \mathbf{R}(A, B)$. In particular, we have $B \circ A : \mathbf{R}(A, B)$. Of course, all elements of $\varphi(A)_n$ and all elements of $\mathbf{R}(A, B)$ arise this way.

Remark 5.6. Note that if L is a nontrivial λ-theory, $\mathbf{R}$ is not a univalent category. To see this, note, for example, that we have an object $X := \langle \pi_1, \pi_2 \rangle : \mathbf{R}$ (corresponding to the type of 'pairs' of λ-terms). Since L_0 is a set, $X = X$ is a proposition. However, X has (at least) two automorphisms:

$$\langle \pi_1, \pi_2 \rangle \quad \text{and} \quad \langle \pi_2, \pi_1 \rangle.$$

These are the identity, and the automorphism (of order 2) that swaps the elements of the pair. To see that these are indeed different morphisms, note that applying them (or their lifted versions) to $(x_{2,1}, x_{2,2})$ gives respectively $x_{2,1}$ and $x_{2,2}$, which are distinct elements by Lemma 4.6.

In the next chapter, the 'universal object' $U : \mathbf{R}$, given by the identity $\lambda x_1, x_1$ (`U`), plays a major role. Note that for all $A : \mathbf{R}$, we have morphisms $A : \mathbf{R}(U, A)$ and $A : \mathbf{R}(A, U)$, which exhibit A as a retract of U (`R_retraction_is_retraction`). Also note that $\varphi(U) = L$.

Note that $\mathbf{R}$ has many (isomorphic, but not equal for nontrivial L) terminal objects, given by $I_c := \lambda x_1, \iota_{0,1}(c) : \mathbf{R}$ for any $c : L_0$ (`R_terminal`). These are terminal because for $f : A \to I_c$, we have $f = I_c \circ f = I_c$. Note that $\varphi(I_c)_n = \{\iota_{0,n}(c)\}$. We will choose $I = \lambda x_1 x_2, x_2 : \mathbf{R}$ as our main example of the terminal object.

We might wonder whether $\mathbf{R}$ also has an initial object O. However, for all $c : L_0$, we would have a constant morphism to the universal object

$$\lambda x_1, \iota_{0,1}(c) : \mathbf{R}(O, U),$$

so if L is nontrivial, $\mathbf{R}$ has no initial object.

$\mathbf{R}$ has binary products with projections and product morphisms (`R_binproducts`)

$$A_1 \times A_2 = \langle p_1, p_2 \rangle, \quad p_i = A_i \circ \pi_i \quad \text{and} \quad \langle f, g \rangle.$$

Recall that for any object A, we have $A = \mathrm{id}_A$, which for $A_1 \times A_2$ is $\langle p_1, p_2 \rangle$ by the universal property of the product, which explains why the product is of this form.

$\mathbf{R}$ also has exponential objects (`R_exponentials`)

$$C^B = \lambda x_1, C \circ x_1 \circ B$$

with evaluation morphism $\epsilon_{BC} : C^B \times B \to C$ given by

$$\epsilon_{BC} = \lambda x_1, C(\pi_1 x_1 (B(\pi_2 x_1))),$$

which is universal because we can lift a morphism $f : \mathbf{R}(A \times B, C)$ to a morphism $\psi(f) : \mathbf{R}(A, C^B)$ given by

$$\psi(f) = \lambda x_1 x_2, f(x_1, x_2).$$

Note that for $g : \mathbf{R}(A, C^B)$, the inverse $\psi^{-1}(g)$ is given by

$$\epsilon \circ \langle g \circ \pi_1, B \circ \pi_2 \rangle = \lambda x_1, g(\pi_1 x_1)(\pi_2 x_1) : \mathbf{R}(A \times B, C).$$

Also note that

$$\psi(\epsilon_{BC}) = \lambda x_1, C \circ x_1 \circ B = C^B = \mathrm{id}_{C^B}.$$

Note that $\varphi(C^B)_0 = \mathbf{R}(B, C)$, as we would expect.

Now, we might wonder whether there exists some $A : \mathbf{R}$ such that $\varphi(A)_0 = \varnothing$. However, for any $c : L_0$, we have $Ac : \varphi(A)_0$, because

$$Ac = (A \circ A)c = A(Ac).$$

Note that we can lift constants from $\varphi(A)_0$ to any $\varphi(A)_n$, so they are all nonempty.

Combining this with the embedding of $\mathbf{R} \hookrightarrow \mathbf{Pshf}_L$, we would expect $\mathbf{R}$ to not have all pullbacks. This is because in $\mathbf{Pshf}_L$, for a cospan $B \xrightarrow{f} A \xleftarrow{g} C$ with $f_n(B_n) \cap g_n(C_n) = \varnothing$ for some n, the pullback Q would have $Q_n = \varnothing$, which could never happen with an object coming from $\mathbf{R}$. Note, however, that this can not be made rigorous, because a fully faithful embedding reflects limits, but does not necessarily preserve them.

However, it is true that $\mathbf{R}$ does not have all pullbacks (if L is nontrivial). Consider for example the following cospan:

$$I \xrightarrow{f} U \xleftarrow{g} I$$

for different $f, g : \mathbf{R}(I, U)$. For example, $f = (\lambda x_1, \iota_{0,1}(\pi_1))$ and $g = (\lambda x_1, \iota_{0,1}(\pi_2))$. Now, take any object $Q : \mathbf{R}$. Note that we have a unique morphism $I : \mathbf{R}(Q, I)$. Then we have the following diagram:

$$\begin{array}{ccc} Q & \xrightarrow{I} & I \\ \downarrow{\scriptstyle I} & & \downarrow{\scriptstyle g} \\ I & \xrightarrow{f} & U \end{array}$$

with

$$f \circ I = f \neq g = g \circ I,$$

so the diagram does not commute, and Q is not a pullback of this cospan.

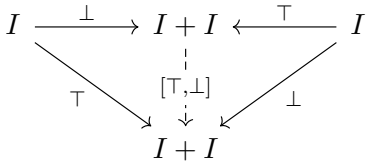
Taylor notes ([Tay86], Section 1.5) that the objects in $\mathbf{R}$ have very strong properties with respect to fixpoints. One of the properties also arises via Lawvere's fixed point theorem

([Law69], page 136): For all $B : \mathbf{R}$, since B^U is a retract of U, every endomorphism $f : B \to B$ has a fixpoint (`fixpoint_is_fixpoint`). That is, there exists $s : I \to B$ such that $f \circ s = s$. Working out the proof even yields an explicit term:

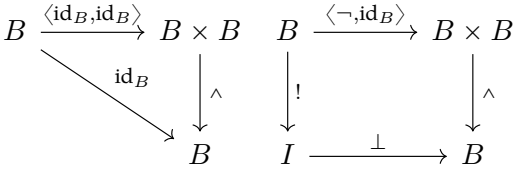
$$s = \lambda i, (\lambda x, f(xx))(\lambda x, f(xx)).$$

Indeed, $s = \lambda i, f((\lambda x, f(xx))(\lambda x, f(xx))) = f \circ s$, and $B \circ s = B \circ f \circ s = s = s \circ I$, so $s : \mathbf{R}(I, B)$.

From this, Taylor deduces ([Tay86], §1.5.12) that $\mathbf{R}$ does not have all coproducts if L is nontrivial, because suppose that $\mathbf{R}$ has all coproducts. Then $B = I + I$ is a 'boolean algebra object': We define $\bot, \top : I \to B$ to be the injections on the left and right components. Since $\mathbf{R}$ is cartesian closed, binary products distribute over binary coproducts, so we have $B \times B \cong ((I \times I) + (I \times I)) + ((I \times I) + (I \times I))$, and note that $I \times I \cong I$. Using the universal property of the coproduct, we can define $\neg : B \to B$ componentwise as $\neg = [\top, \bot]$, the coproduct arrow



Note that by the definition of $\neg$, $\neg \circ \bot = \top$. Similarly, we define $\wedge, \vee : B \times B \to B$ as $\wedge = [[\bot, \bot], [\bot, \top]]$ and $\vee = [[\bot, \top], [\top, \top]]$. Using the same universal property, we can also verify some properties of $\bot, \top, \wedge$ and $\neg$, like the fact that the following diagrams commute:



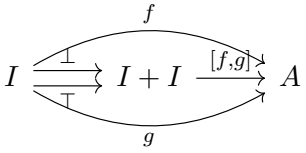
for $! : B \to I$ the terminal projection. We do this by checking, for example, that $\mathrm{id}_B \circ \top = \wedge \circ \langle \mathrm{id}_B, \mathrm{id}_B \rangle \circ \top$ and $\mathrm{id}_B \circ \bot = \wedge \circ \langle \mathrm{id}_B, \mathrm{id}_B \rangle \circ \bot$. Now, as mentioned above, every endomorphism has a fixed point. In particular, we have some $\star : I \to B$ such that $\neg \circ \star = \star$. Now, note that

$$\star = \wedge \circ \langle \mathrm{id}_B, \mathrm{id}_B \rangle \circ \star = \wedge \circ \langle \star, \star \rangle = \wedge \circ \langle \star, \neg \circ \star \rangle = \wedge \circ \langle \mathrm{id}_B, \neg \rangle \circ \star = \bot \circ ! \circ \star = \bot$$

and then

$$\bot = \star = \neg \circ \star = \neg \circ \bot = \top.$$

Now, for any object $A : \mathbf{R}$ and any two global elements $f, g : I \to A$, we have the following diagram:



and we have

$$f = [f, g] \circ \bot = [f, g] \circ \top = g.$$

In particular, for any two objects $A, A' : \mathbf{R}$, since we have $A \circ A' : \mathbf{R}(A', A)$, so $\mathbf{R}(A', A)$ is nonempty, we have

$$\mathbf{R}(A', A) \simeq \mathbf{R}(I \times A', A) \simeq \mathbf{R}(I, A^{A'}) \simeq \{\star\},$$

so $\mathbf{R}$ is the trivial category and L is trivial.

5.3 Scott's Representation Theorem

The correspondence in Section 5.1 between simply-typed λ-calculi and cartesian closed categories raises a question whether such a correspondence also exists for untyped λ-calculi. Definition 4.51 shows that in fairly general circumstances we can take one object c in a category C and consider the morphisms $t : C(c^n, c)$ as terms in an untyped λ-calculus. Hyland calls this the 'endomorphism theory' of c.

Remark 5.7. To construct a *simply typed* λ-calculus from a category, we just need a cartesian closed category. In a simply typed λ-calculus, there is a lot of restriction on which terms we can apply to each other. A term of type $A \to B$ can only be applied to a term of type A, which gives a term of type B. In particular, a term can be applied only finitely many times to other terms, and every time, the result has a different type.

On the other hand, for an *untyped* λ-calculus, we need a cartesian closed category with a 'reflexive object'. This is because in the untyped λ-calculus, we can apply arbitrary terms to each other. For example, we can apply the term $(\lambda x_1, x_1 x_1)$ to itself, which would not be typable in the simply typed λ-calculus. Suppose that we have a category C and an object U such that the morphisms $C(U^n, U)$ give the untyped λ-terms in n free variables, for all n. Now, given two terms $f, g : C(U^n, U)$, for the application fg, we need to consider f as a morphism in $C(U^n, U^U)$. We can do this by postcomposing with a morphism $\varphi : C(U, U^U)$. On the other hand, if $n > 0$, then $(\lambda x_n, f)$ is a morphism in $C(U^{n-1}, U^U)$, but it is a term in $n-1$ free variables, so it should be in $C(U^{n-1}, U)$. For this, we postcompose with a morphism $\psi : C(U^U, U)$. Now, for our untyped λ-calculus to have β-equality, we need $\psi \cdot \varphi = \mathrm{id}_{U^U}$, which means that the exponential U^U of U is a retract of U. This is exactly what it means for U to be a *reflexive object*: an object U in a cartesian closed category, that has a retraction onto its 'function space' U^U. Note that if we want our λ-theory to also have η-equality, the retraction must be an isomorphism.

Note that **Set** is a cartesian closed category, but that for sets X and Y, the function space X^Y has cardinality $|X|^{|Y|}$, and therefore U^U cannot be a retract of U, unless $U = \{\star\}$, in which case we have a very trivial λ-calculus: $\mathbf{Set}(U^n, U) = \{\star\}$.

During the 1960s, computer scientists sought for nontrivial examples of reflexive objects, there is a quote by Dana Scott that "Lambda-calculus has no mathematical models!"[Sco16]. However, in 1969, the same Dana Scott discovered that the category of (continuous) lattices with (Scott-)continuous functions between them has a nontrivial reflexive object, with the retraction onto the function spaces being even an isomorphism. Such a reflexive object D_∞ is obtained by starting with an arbitrary lattice D_0, iteratively taking $D_{n+1} = D_n^{D_n}$, with a retraction (in the 'wrong' direction) $r : D_n^{D_n} \to D_n$, and then passing to the limit $D_\infty = \lim_{\leftarrow} D_n$ (for the main result, see Theorem 4.4 in [Sco72], page 127).

Since Lambek showed an equivalence between simply typed λ-calculi and cartesian closed categories, and since we have a construction for an untyped λ-calculus from a cartesian closed category with a reflexive object, we can wonder whether this construction constitutes contitutes an equivalence between untyped λ-calculi and some class of categories. This question finds a partial answer in the following theorem, originally proved in a very syntactical way by Dana Scott (see [SH80], page 418).

Theorem 5.8 (`representation_theorem_iso`)**.** *We can obtain every untyped λ-calculus as the endomorphism theory of some object in some category.*

Proof. Let L be a λ-theory. Scott considers its category of retracts **R**, with 'universal object' U.

Note that U^U is a retract of U, so U is a reflexive object.

Therefore, $E(U)$, the endomorphism theory of U, has a λ-theory structure. Note that the finite powers of U in **R** are given by $U^0 = I$ and $U^{n+1} = U^n \times U$.

We have $E(U)_n = \mathbf{R}(U^n, U) = \{f : L_0 \mid U \circ f \circ U^n = f\}$. The variables of $E(U)$ are the projections $\pi_{n,i}$ of U^n. The substitution is given by composition with the product morphism:

$$f \bullet g = f \circ \langle\langle\langle I, g_1\rangle, \dots\rangle, g_n\rangle.$$

We have $U^U = \lambda f, U \circ f \circ U = \lambda x_1 x_2, x_1 x_2$. Using the equivalence $\mathbf{R}(U^n \times U, U) \simeq \mathbf{R}(U^n, U^U)$ and the retraction $U^U : U \to U^U$, the abstraction and application λ and ρ are given by

$$\lambda(f) = \lambda x_1 x_2, \iota_{0,2}(f)(x_1, x_2), \quad \rho(g) = \lambda x_1, \iota_{0,1}(g)(\pi_1 x_1)(\pi_2 x_1).$$

for $f : \mathbf{R}(U^{n+1}, U)$ and $g : \mathbf{R}(U^n, U)$.

Now, we have bijections $\psi_0 : E(U)_0 \xrightarrow{\sim} L_0$, given by

$$\psi_0(f) = f(\lambda x_1, x_1) \quad \text{and} \quad \psi_0^{-1}(g) = \lambda x_1, \iota_{0,1}(g).$$

We can extend this to any n, by reducing any term to a constant by repeatedly using λ, then applying the bijection, and then lifting it again using ρ. Explicitly, we obtain

$$\psi_n(f) = \iota_{0,n}(f)(x_i)_i, \quad \text{and} \quad \psi_n^{-1}(g) = \lambda x_1, g \bullet (\pi_{n,i} x_1)_i.$$

It is not hard to verify that this is indeed a bijection, using at one point the fact that $f : \mathbf{R}(U^n, U)$ is defined by $f \circ U^n = f$, for

$$U^n = \langle \pi_{n,i} \rangle_i.$$

It is also pretty straightforward to check that

$$\begin{aligned} \psi(\pi_{n,i}) &= x_i, & \psi(f) \bullet (\psi(g_i))_i &= \psi(f \bullet g), \\ \psi(\lambda(h)) &= \lambda(\psi(h)), & \psi(\rho(h')) &= \rho(\psi(h')) \end{aligned}$$

for $f : E(U)_m$, $g : E(U)_n^m$, $h : E(U)_{n+1}$ and $h' : E(U)_n$. Therefore, ψ is an isomorphism of λ-theories. □

5.4 The Taylor Fibration

In his dissertation, Paul Taylor shows that **R** is not only cartesian closed, but also *relatively cartesian closed*. Recall we have chosen to interpret **R** as the category $\overline{C}_{L_0}$, constructed using idempotents, because both Scott's and Taylor's proof are very syntactical in nature.

In Section 2.7, we studied internal and external representations of families of objects in a category and how they behaved under substitutions (pullbacks). This was to arrive at a definition for dependent products and sums, as the right and left adjoints to the pullback (or substitution) functor $\alpha^* : (C \downarrow A) \to (C \downarrow B)$ along some morphism $\alpha : C(B, A)$.

Now, some categories are not locally cartesian closed. That is: not all pullback/substitution functors α^* exist or have a right adjoint. For example, **R** does not have all pullbacks, so the substitution functors do not always exist. In such a category C, the functor $C^2 \to C$ is not a fibration. One way to look at this, is that in such a category not every morphism $X \to A$ represents a family of objects. In such a category, we can carefully choose a subset of the morphisms to represent our indexed families. We will call a morphism that we choose to represent an indexed family a *display map*. In most cases, we have quite a bit of choice how big we want our subtype of display maps to be. However, to make sure that indexed families are well-behaved, the subtype of display maps needs to have some properties:

1. The pullback of a display map along any morphism exists and is a display map.
2. The composite of two display maps is a display map.
3. C has a terminal object and any terminal projection is a display map.

Remark 5.9. A 'maximal' example of a subtype of display maps is, for example, in the category **Set**, where we can take our subtype of display maps to equal the full type of all morphisms of C.

Remark 5.10. A 'minimal' example of a subtype of display maps is the subtype of (maps isomorphic to) product projections in a category with finite products. In this case, all indexed families are constant, and then dependent sums and products become binary products and exponential objects.

Now, let C be a category with a subtype of display maps $D \subseteq C$. We will denote the subtype of display maps from X to A with $D(X, A) \subseteq C(X, A)$. For any $A : C$, we define the category $(C \downarrow_D A)$ as a full subcategory of the slice category $(C \downarrow A)$, with as objects the display maps $f : D(X, A)$. The morphisms between two objects of $(C \downarrow_D A)$ are still all the morphisms of $(C \downarrow A)$ (i.e. the morphisms of C that commute with the display maps).

Note that for the terminal object $I : C$, $(C \downarrow_D I)$ is still equivalent to C, since every terminal projection is a display map.

Also note that since the pullback of display maps against any map exist (and are display maps again), we get a pullback functor $\alpha^* : (C \downarrow_D A) \to (C \downarrow_D B)$. This is the restriction of the pullback functor $\alpha^* : (C \downarrow A) \to (C \downarrow B)$, if it exists.

Note that since composing two display maps gives a display map again, and since the dependent sum is given by postcomposition, α^* has a left adjoint for all display maps α. That is: the fiber categories $(C \downarrow_D A)$ have dependent sums over display maps.

The question whether the fiber categories $(C \downarrow_D A)$ also have dependent products over display maps, brings us to the definition of relative cartesian closedness.

Definition 5.11. A category C is *cartesian closed relative* to a class of display maps D, if the substitution functors α^* along display maps have right adjoints.

Now, analogously to Lemma 2.33, Taylor shows:

Lemma 5.12. *If a category C is cartesian closed relative to a class of display maps D, then the fiber categories $(C \downarrow_D A)$ are cartesian closed and the substitution functors α^* preserve this structure.*

Proof. Taylor proves this in a series of lemmas leading up to §4.3.7 in [Tay86]. It is also proved as Proposition 6 of [HP89]. □

5.4.1 Taylor's proof

As Hyland remarks, Taylor shows that **R** is relatively cartesian closed using a very syntactical argument, in the spirit of Scott.

He starts out with a kind of 'external' representation of indexed families $(X_a)_a$ in **R**. He denotes these as 'functions' $A \to \mathbf{R}_0$. They are the elements $X : L_0$ with

$$X \circ A = X \quad \text{and} \quad (\lambda x_1, (Xx_1) \circ (Xx_1)) = X.$$

These X form a category $\mathbf{R}^A$, with the morphisms in $\mathbf{R}^A(X, Y)$ given by $f : L_0$ with $f \circ A = f$ and $(\lambda x_1, (Yx_1) \circ (fx_1) \circ (Xx_1)) = f$. The identity on X is given by X, and the composition is given by $f \cdot g = \lambda x_1, gx_1 \circ fx_1$.

Taylor shows that these categories $\mathbf{R}^A$ are cartesian closed. He notes that assigning these categories $\mathbf{R}^A$ to objects $A : \mathbf{R}$ again constitutes a contravariant (pseudo)functor $\mathbf{R}^{\mathrm{op}} \to \mathbf{Cat}$, sending morphisms $A \to B$ to precomposition functors $\mathbf{R}^B \to \mathbf{R}^A$.

For $A : \mathbf{R}$, we introduce the combinator

$$\sum_A = \lambda x_1 x_2, (A(\pi_1 x_2), x_1(\pi_1 x_2)(\pi_2 x_2)).$$

For $A : \mathbf{Set}$, we have an equivalence between the elements of $\mathbf{Set}^A$ and the elements of the fiber $(\mathbf{Set} \downarrow A)$ of the fibration $\mathbf{Set}^2 \to \mathbf{Set}$. Now, for $A : \mathbf{R}$, $\sum_A$ gives a functor from $\mathbf{R}^A$ to $(\mathbf{R} \downarrow A)$. Note that it sends objects $X : \mathbf{R}^A$ to $\sum_A X$, together with a projection morphism $p_X = A \circ \pi_1 : \sum_A X \to A$. Note that with the embedding in Definition 5.3 into $\mathbf{Pshf}_L$, we can consider $\sum_A X$ as consisting of pairs (a, x) with $a : A$ and $x : Xa$, which is exactly what we would expect from a dependent sum.

Now, $\sum_A$ is fully faithful: given a morphism $g : \mathbf{R}\,(\sum_A X, \sum_A Y)$, we take

$$\psi(g) = \lambda x_1 x_2, \pi_2(g(x_1, x_2)).$$

For verifying that $\psi(g)$ is indeed a morphism in $\mathbf{R}^A(X, Y)$, it helps to recall that $g \circ \sum_A X = g = \sum_A Y \circ g$ and $p_Y \circ g = p_X$. Since $\psi(\sum_A f) = f$ for all $f : \mathbf{R}^A(X, Y)$ and $\sum_A \psi(g) = g$ for all $g : \mathbf{R}(\sum_A X, \sum_A Y)$, $\sum_A$ is indeed fully faithful.

On the other hand, $\sum_A$ is generally not surjective. However, we can choose our subtype D of display maps in such a way that the restriction $\mathbf{R}^A \to (C \downarrow_D A)$ becomes essentially surjective.

Definition 5.13. For $\mathbf{R}$, we will consider a morphism $f : X \to A$ to be a display map if we have some $Y : \mathbf{R}^A$ like above, and some isomorphism $g : (X, f) \xrightarrow{\sim} (\sum_A Y, p_Y)$ in $(\mathbf{R} \downarrow A)$.

Remark 5.14. Hyland actually gives a different characterization of Taylor's display maps. He claims that Taylor takes the display maps $X \to A$ to be the retracts in $(\mathbf{R} \downarrow A)$ of $p_1 : A \times U \to A$, the projection onto the first coordinate. The two characterizations are equivalent:

Given $Y : \mathbf{R}^A$, intuitively Ya is a retract of U for all a. Concretely, both morphisms $r : A \times U \to \sum_A Y$ and $s : \sum_A Y \to A \times U$ are given by $\sum_A Y$ and these commute with the projection to A. Therefore, if we have an isomorphism $g : X \xrightarrow{\sim} \sum_A Y$ in $(\mathbf{R} \downarrow A)$, then $\sum_A Y \cdot g^{-1}$ and $g \cdot \sum_A Y$ make X into a retract of $A \times U$.

Conversely, given a retraction $r : A \times U \to X$ and section $s : X \to A \times U$ in $(\mathbf{R} \downarrow A)$, we have

$$Y = \lambda x_1 x_2, \pi_2(s(r(x_1, x_2))) : \mathbf{R}^A.$$

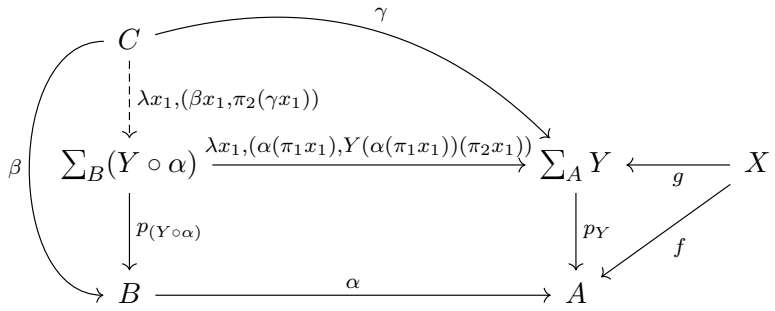
Using the properties of r and s and the definition of $\sum_A Y$, one can show that r gives a morphism $\sum_A Y \to X$ and s gives a morphism $X \to \sum_A Y$, and that $r \circ s = \mathrm{id}_{\sum_A Y}$. Combined with the fact that $s \circ r = \mathrm{id}_X$, this shows that X is isomorphic to $\sum_A Y$.

Remark 5.15. Recall that if L is nontrivial, $\mathbf{R}$ is not univalent, and the existence of $Y : \mathbf{R}^A$ with the isomorphism $g : X \xrightarrow{\sim} \sum_A Y$ is not a proposition. Take, for example, $Y : \mathbf{R}^I$ given by $(\lambda x_1, U \times U)$. Recall that $(\mathbf{R} \downarrow I)$ is equivalent to $\mathbf{R}$. Under this equivalence $\sum_I Y$ is equivalent to $U \times U$. As mentioned before, $\sum_I Y$ has (at least) two distinct isomorphisms to itself: the identity and the isomorphism that swaps both sides of the product.

In a similar way, being a retract of $A \times U$ is not a proposition. Therefore, if we want the class of display maps to really be a subclass of the class of morphisms, we need the existence of $Y : \mathbf{R}^A$ and the isomorphism $g : X \xrightarrow{\sim} \sum_A Y$, or the existence of the retraction $A \times U \to X$, to mean *mere existence* in this case. This means that we cannot use these in constructions, except for mere propositions.

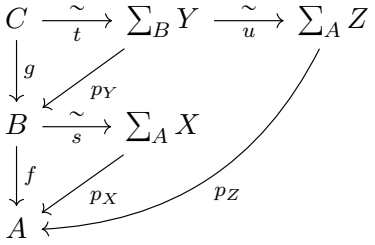
Taylor shows that these display maps indeed satisfy the three properties mentioned above. However, in univalent foundations, there are some subtleties in the case of pullbacks:

1. Given a display map $(X, f) : (\mathbf{R} \downarrow_D A)$ and a morphism $\alpha : \mathbf{R}(B, A)$. Recall that this means that there merely exists a $Y : \mathbf{R}^A$ and an isomorphism $X \xrightarrow{\sim} \sum_A Y$ in $(\mathbf{R} \downarrow A)$. Therefore, we have mere existence of a pullback $\sum_B (Y \circ \alpha)$ of $\sum_A Y$ along α. For all $C : \mathbf{R}$ with $\beta : \mathbf{R}(C, B)$ and $\gamma : \mathbf{R}(C, \sum_A Y)$ that make the square commute, we have the morphism $\lambda x_1, (\beta x_1, \pi_2(\gamma x_1)) : \mathbf{R}(C, \sum_B (Y \circ \alpha))$.



By the isomorphism between X and $\sum_A Y$, $\sum_B(Y \circ \alpha)$ is also a pullback of X: by postcomposing with g and its inverse, we see that morphisms from $\sum_B(Y \circ A)$ and C to X are equivalent to morphisms to $\sum_A Y$.

2. Given display maps $f : B \to A$ and $g : C \to B$. We have $X : \mathbf{R}^A$ and $Y : \mathbf{R}^B$ with isomorphisms $s : B \xrightarrow{\sim} \sum_A X$ and $t : C \xrightarrow{\sim} \sum_B Y$. We need to show that $f \circ g : C \to A$ is also a display map.



Intuitively, $\sum_B Y \cong \sum_A Z$ represents

$$\sum_{b:\sum_{a:A} Xa} Yb \cong \sum_{a:A} \sum_{x:Xa} Y(a,x),$$

by the isomorphism that relates $((a,x),y)$ to $(a,(x,y))$. Therefore, consider

$$Z = \lambda x_1 x_2, (Xx_1(\pi_1 x_2), Y(s^{-1}(x_1, \pi_1 x_2))(\pi_2 x_2)) : \mathbf{R}^A.$$

we have an isomorphism $u : \sum_B Y \xrightarrow{\sim} \sum_A Z$ given by:

$$u = \lambda x_1, (Aa, (Xax, Yby))$$

with

$$b = \pi_1 x_1; \quad a = \pi_1(sb); \quad x = \pi_2(sb); \quad y = \pi_2 x_1,$$

and

$$u^{-1} = \lambda x_1, (s^{-1}(Aa, Xax), Yby)$$

with

$$a = \pi_1 x_1; \quad x = \pi_1(\pi_2 x_1); \quad b = s^{-1}(a,x); \quad y = \pi_2(\pi_2 x_1)$$

and then $u \circ t$ is an isomorphism in $(\mathbf{R} \downarrow A)$ between $(C, f \circ g)$ and $(\sum_A Z, p_Z)$.

3. Let $B \to I$ be a terminal projection. Consider $Y = (\lambda x_1, B) : \mathbf{R}^I$. We have

$$\sum_I Y = \lambda x_1, ((\lambda x_2, x_2), B(\pi_2 x_1)) = \langle I \circ \pi_1, B \circ \pi_2 \rangle = I \times B,$$

which is isomorphic to B.

Remark 5.16. There is an awful lot of properties to verify here (e.g. that some terms are elements of some $\mathbf{R}^A$, that others are morphisms in some $\mathbf{R}(\sum_A X, \sum_B Y)$, that some diagrams commute), but most of it boils down to writing down the equations that should hold, reducing them via β-equality, and then using the facts that objects $A : \mathbf{R}$ are idempotent, that morphisms $f : \mathbf{R}(A, B)$ equal $B \circ f \circ A$ and that objects $X : \mathbf{R}^A$ satisfy $X \circ A = X$ and $Xa(Xax) = Xax$ for all a and x. Note that for a morphism $f : \sum_A X \to B$,

$$f(Ax_1, x_2) = f(x_1, x_2) = f(x_1, Xx_1x_2),$$

so f 'absorbs' such instances of A and X, so to speak.

Remark 5.17. To talk about 'relatively cartesian closed', we need a pullback functor $\alpha^* : (\mathbf{R} \downarrow_D A) \to (\mathbf{R} \downarrow_D B)$ for all (display maps) $\alpha : B \to A$. However, as remarked before, since the definition of display maps involves a propositional truncation, we only have mere existence of pullbacks, and to pass from the statement "for all f, there exists a pullback $\alpha^* f$" to the statement "there exists a function that sends every morphism $f : (\mathbf{R} \downarrow_D A)$ to its pullback $\alpha^* f : (\mathbf{R} \downarrow_D B)$", we need to assume the axiom of choice.

However, note that we have a weak equivalence between strict categories (categories in which the type of objects is a homotopy set) $\sum_A : \mathbf{R}^A \xrightarrow{\sim} (\mathbf{R} \downarrow_D A)$. If we assume the axiom of choice, $\sum_A$ becomes an equivalence of categories (see bullet (ii) in the introduction of Chapter 9 of [Uni13]) and then our pullback functor is given by

$$\sum_A^{-1} \bullet (\alpha \cdot -) \bullet \sum_B,$$

where $(\alpha \cdot -) : \mathbf{R}^A \to \mathbf{R}^B$ denotes the functor given by precomposition with α.

This brings us to the main theorem of this section ([Tay86], 5.1.8).

Theorem 5.18. *If we assume the axiom of choice,* $\mathbf{R}$ *is cartesian closed, relative to the given class of display maps.*

Proof. Let $\alpha : B \to A$ be a display map. We have mere existence of some $X : \mathbf{R}^A$ and an isomorphism $g : (X, \alpha) \xrightarrow{\sim} (\sum B, p_X)$. We need to show that there is a right adjoint to the pullback functor α^*. Since under assumption of the axiom of choice, α^* is defined as a composition of a precomposition functor with two equivalences of categories, this is equivalent to showing that there is a right adjoint to the precomposition functor $(\alpha \cdot -) : \mathbf{R}^A \to \mathbf{R}^B$ (note that $\alpha \cdot -$ sends f to $f \circ \alpha$).

Now, intuitively, any $Y : \mathbf{R}^B$ denotes an indexed family $((Y(a, b))_{b:Xa})_{a:A}$. We want the right adjoint $\alpha_* Y$ to be the indexed family of dependent products $(\prod_{b:B_a} Y(a, b))_a$. An element f of the dependent product $\alpha_* Ya$ is then a function from Xa, that satisfies $fb : Y(a, b)$ for all $b : Xa$. Therefore, for all $a : A$ and all $f : L_0$, we want $(\alpha_* Ya)f = f$ to mean "$f \circ Xa = f$ and, $fb : Y(a, b)$ for all $b : Xa$". We encode these two parts as

$$\lambda x_1, x_1 \circ Xa \quad \text{and} \quad \lambda x_1 x_2, Y(g^{-1}(a, x_2))(x_1 x_2).$$

It turns out that these are idempotents and they commute, so we can compose them into one object

$$\alpha_* Ya = \lambda x_1 x_2, Y(g^{-1}(a, x_2))(x_1(Xax_2)) : \mathbf{R}$$

and we define the combinator

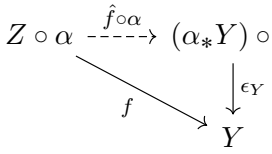
$$\alpha_* = \lambda x_1 x_2 x_3 x_4, x_1(g^{-1}(x_2, x_4))(x_3(Xx_2x_4)).$$

Now, applying α_* to both objects and morphisms in $\mathbf{R}^A$ gives a functor $\alpha_* : \mathbf{R}^B \to \mathbf{R}^A$.

Note that for all $Y : \mathbf{R}^B$ and all b, we have a morphism $(\lambda x_1, x_1(\pi_2(gb))) : \mathbf{R}(\alpha_* Y(\alpha b), Yb)$. Making this parametric in b gives us a counit

$$\epsilon_Y = \lambda x_1 x_2, Y x_1(x_2(\pi_2(g x_1))) : \mathbf{R}^B((\alpha_* Y) \circ \alpha, Y).$$

To show that $(\alpha \cdot -) \dashv \alpha_*$, we need to show that for all $Y : \mathbf{R}^B$, $(\alpha_* Y, \epsilon_Y)$ is a universal arrow from $(\alpha \cdot -)$ to Y. Recall the following diagram:



Now, suppose that we have some $Z : \mathbf{R}^A$ and some $f : \mathbf{R}^B(Z \circ \alpha, Y)$. We define

$$\hat{f} = \lambda x_1 x_2 x_3, f(g^{-1}(x_1, x_3))(Z x_1 x_2) : \mathbf{R}^A(Z, \alpha_* Y)$$

and we have $((\hat{f}) \circ \alpha) \cdot \epsilon_Y = f$. Now, suppose that we have some (other) $\hat{f}' : \mathbf{R}^A(Z, \alpha_*(Y))$ such that $(\hat{f}' \circ \alpha) \cdot \epsilon_Y = f$. Then substituting $(\hat{f}' \circ \alpha) \cdot \epsilon_Y$ for f in $\hat{f}$ yields

$$\hat{f} = \lambda x_1, (\alpha_* Y x_1) \circ (\hat{f}' x_1) = \hat{f}',$$

so $\hat{f}$ is unique and $(\alpha_* Y, \epsilon_Y)$ is a universal arrow, which concludes the proof that $(\alpha \cdot -) \dashv \alpha_*$ and we see that $\mathbf{R}$ is cartesian closed relative to the given class of display maps. □

Remark 5.19. Recall that in the definition of α_*, we composed commuting idempotents

$$\lambda x_1 x_2, x_1(X a x_2) \quad \text{and} \quad \lambda x_1 x_2, Y(a, x_2)(x_1 x_2).$$

In his PhD thesis, Taylor instead composes the idempotents

$$\lambda x_1, x_1(X a x_2) \quad \text{and} \quad \lambda x_1 x_2, Y x_2(x_1 x_2).$$

Note that in the first term, x_2 plays the role of an element of Xa, whereas in the second term, it plays the role of an element of B. Of course, if we worked in **Set**, we would indeed have $Xa \subseteq \bigsqcup_{a:A} X_a \cong B$. However, in $\mathbf{R}$, the 'terms' of $B \cong \sum_A X$ look like pairs (a, x) with $a : A$ and $x : Xa$, so we cannot consider terms of Xa to be terms of B.

Therefore, the idempotents that he uses do not commute, and the resulting term

$$\lambda x_1 x_2, Y(B x_2)(x_1(X a x_2))$$

(notice the redundant usage of B) is not idempotent.

5.5 The λ-calculus is Algebraic

As we mentioned in Remark 4.32, using tools from universal algebra, there are two approaches to study λ-calculus-like structures. One can either study λ-theories or algebras for the algebraic theory Λ. We will see that Hyland shows that these are equivalent.

In his proof, he refers to the 2002 paper 'The lambda calculus is algebraic' by Peter Selinger [SEL02]. We will not study this paper in detail, but it is interesting to have a quick look at it, because it seems that Hyland took some inspiration from it.

Selinger studies two kinds of objects which he calls λ-algebras and λ-theories:

Definition 5.20. A *combinatory algebra* is a set A, together with a binary (application) operation $(a_1, a_2) \mapsto a_1 a_2$, and distinguished elements $k, s : A$ such that for all $a_1, a_2, a_3 : A$,

$$ka_1a_2 = a_1 \quad \text{and} \quad sa_1a_2a_3 = a_1a_3(a_2a_3).$$

A λ-*algebra* is a combinatory algebra which 'behaves nicely' in some sense[1].

For a set C and a countable set V, we define Λ_C to be the inductive type with constructors

$$\mathtt{Var}(x), \quad \mathtt{App}(f, g), \quad \mathtt{Abs}(x, f) \quad \text{and} \quad \mathtt{Con}(c)$$

for $x : V$, $f, g : \Lambda_C$ and $c : C$.

Definition 5.21. A λ-*theory* is an equivalence relation '$=$' on Λ_C for some C, containing α- and β-equality and closed under application and abstraction.

Selinger shows in his paper that the categories of λ-algebras and λ-theories are equivalent. As we will see in the next chapter, Hyland shows that his own notions of Λ-algebras and λ-theories are equivalent.

Now, it is not hard to show that Hyland's and Selinger's notions of λ-theory are equivalent: If we have one of Selinger's λ-theories with a countable set of variables, enumerate the variables as $x_1, x_2, \ldots$ and let L_n be the set of terms with only the free variables $x_1, \ldots, x_n$. Then, if we quotient by the equivalence relation, we get one of Hyland's λ-theories. Conversely, if we start with one of Hyland's λ-theories L, we can consider L_n as a subset of L_{n+1} by sending $x_{n,i}$ to $x_{n+1,i}$. If we take the union $\bar{L} = \bigcup_n L_n$ (officially, this is a directed limit), we have a function $f : \Lambda_{\bar{L}} \to \bar{L}$, because the constants of $\Lambda_{\bar{L}}$ are exactly the terms of $\bar{L}$ and because λ-theories in Hyland's sense support the operations of the λ-calculus (see Subsection 4.4.1). This gives an equivalence relation on $\Lambda_{\bar{L}}$ by taking $a \mathrel{'='} b$ if $f(a) = f(b)$.

Note, however, that is very hard to show that Hyland's notion of Λ-algebra and Selinger's notion of λ-algebra are equivalent. If we start with a Λ-algebra A, this is already a combinatory algebra, because Λ contains the S and K combinators and their image are the designated elements s and k. However, if we start with a λ-algebra and want to turn this into a Λ-algebra, we need to give a t-action for every $t : \Lambda$, not just for S and K. Therefore, we need to show that any Λ-term can be expressed in terms of K, S and application. However, at that point we are already halfway through a proof that any Λ-algebra can be given a λ-theory structure.

[1] We can interpret terms of combinatory algebra (abstract terms involving just application, the constants S and K, constants from A and some variables) as functions on A. We can also interpret them as λ-terms with constants in A, by sending S and K to the S and K combinators. Then 'behaving nicely' means that if the λ-terms of two such terms are equal, then their functions on A are equal as well.

Chapter 6

Hyland's Paper

6.1 Scott's Representation Theorem

Now, Scott's representation theorem, which asks whether we can represent any λ-theory as the endomorphism theory of some reflexive object in some cartesian closed category, can be answered using the presheaf category.

Theorem 6.1 (`representation_theorem_iso`)**.** *Any λ-theory L is isomorphic to the endomorphism λ-theory $E_{\mathbf{Pshf}_L}(L)$, with L viewed as a presheaf.*

Proof. First of all, remember that L is indeed exponentiable and that $L^L = A(L,1)$. Now, since L is a λ-theory, we have sequences of functions back and forth $\lambda_n : A(L,1)_n \to L_n$ and $\rho_n : L_n \to A(L,1)_n$. These commute with the L-actions, so they constitute presheaf morphisms and $E(L)$ is indeed a λ-theory.

Remark 4.59 gives a sequence of bijections $\varphi_n : \mathbf{Pshf}_L(L^n, L) \cong L_n$ for all n, sending $F : \mathbf{Pshf}_L(L^n, L)$ to $F(x_1, \ldots, x_n)$, and conversely sending $s : L_n$ to $((t_1, \ldots, t_n) \mapsto s\bullet(t_1, \ldots, t_n))$. It considers λ-terms in n variables as n-ary functions on the λ-calculus. Therefore, it should come as no surprise that φ preserves the x_i, $\bullet$, ρ and λ, which makes it into an isomorphism of λ-theories and this concludes the proof. □

So Hyland shows that the representation theorem follows largely from the fact that the functions from L_n to itself can be represented by L_{n+1}, together with the Yoneda lemma for the Lawvere theory associated to L (Remark 4.59).

This proof is particularly nice, because it does not require β- or η-equality, but if L has β- or η-equality, then we can immediately see (even without the isomorphism from the theorem) that the endomorphism theory also has this property.

Remark 6.2. Actually, in this thesis we stumbled upon another category with a reflexive object that exhibits L as its endomorphism theory, provided that L has β-equality: In Appendix B, we see that the Lawvere theory C associated to L is a weak cartesian closed category, in which $1^1 = 1$. It is then easy to see that the endomorphism theory that we obtain from this reflexive object 1 is exactly equal to L.

Note, however, that this is technically not an answer to Scott's original theorem, because the original formulation asks for a cartesian closed category with a reflexive object, whereas here we only have a reflexive object in a weak cartesian closed category. Even so, it is an answer to the core of the question, which is about whether we can represent any λ-theory as the sets of morphisms $C(U^n, U)$ for some $U : C$.

6.2 Relations Between the Categories

As mentioned before, we have a fully faithful embedding of $\mathbf{R}$ into $\mathbf{Pshf}_L$.

Also, we have a functor from the Lawvere Theory **L** (see Lemma A.6) associated to L into **R**, sending $n : \mathbf{L}$ to $U^n : \mathbf{R}$. By Scott's representation theorem, this functor can map morphisms as follows:

$$\mathbf{L}(m,n) = L_m^n \cong \mathbf{R}(U^m, U)^n \cong \mathbf{R}(U^m, U^n),$$

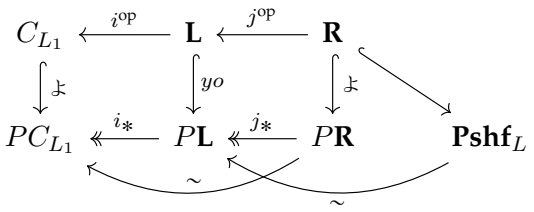
which immediately shows that this functor is a fully faithful embedding.

Note that if we consider L_1 as a monoid, with operation $\bullet$ and unit x_1, and if we consider this monoid as a category, we can embed this (fully faithfully) into **L**. This gives the following sequence of embeddings:

$$L_1 \hookrightarrow \mathbf{L} \hookrightarrow \mathbf{R} \hookrightarrow \mathbf{Pshf}_L$$

Note that the composition of the first two morphisms sends the object of the category L_1 to $(\lambda x_1, x_1) : \mathbf{R}$, which is exactly the usual embedding of L_1 into its Karoubi envelope.

Write $i : L_1^{\mathrm{op}} \hookrightarrow \mathbf{L}^{\mathrm{op}}$ and $j : \mathbf{L}^{\mathrm{op}} \hookrightarrow \mathbf{R}^{\mathrm{op}}$ for the embeddings on the opposite categories. By Corollary 2.42, the first two functors in this sequence essentially yield surjective functors on the presheaf categories:



The two equivalences in this diagram are due to Lemma A.8 and Corollary 2.81.

Lemma 6.3 (`adjoint_equivalence_1_from_comp`)**.** *The precomposition functors i_* and j_* are adjoint equivalences.*

Proof. We will show that j_* is an adjoint equivalence. From the 'two out of three' property, it follows then that i_* is also an adjoint equivalence.

Lemma 2.40 shows that $\eta : \mathrm{id}_{P\mathbf{L}} \Rightarrow \mathrm{Lan}_j \bullet j_*$ is a natural isomorphism. To complete the proof that j_* is an adjoint equivalence, we just have to show that $\epsilon : j_* \bullet \mathrm{Lan}_j \Rightarrow \mathrm{id}_{P\mathbf{L}}$ is a natural isomorphism as well.

To this end, take $F : P\mathbf{R}$. By Lemma 2.40, we have the isomorphism

$$\eta^{-1}_{j_*F} : j_*(\mathrm{Lan}_j(j_*F)) \cong j_*F.$$

Since functors preserve isomorphisms, we have $i_*(j_*(\mathrm{Lan}_j(j_*F))) \cong i_*(j_*F)$. However, since $j_* \bullet i_*$ is an equivalence and in particular fully faithful, this corresponds to an isomorphism

$$\mathrm{Lan}_j(j_*F) \cong F.$$

Now we only need to prove that its morphism is equal to ϵ_F. Equivalently, we need to prove that $i_*(j_*\epsilon)$ is equal to the morphism $i_*\eta^{-1}_{j_*F}$, or that $j_*\epsilon$ equals $\eta^{-1}_{j_*F}$.

Take $n : \mathbf{L}$. We need to show that $\epsilon_F(j(n)) = \eta^{-1}_{j_*F}(n)$ as functions from $(\mathrm{Lan}_j(j_*F))(j(n))$ to $F(j(n))$. Note that the diagram for $(\mathrm{Lan}_j(j_*F))(j(n))$ consists of $F(j(m))$ for all $f : \mathbf{R}(j(m), j(n))$. Now, as it turns out, both $\epsilon_F(j(n))$ and $\eta^{-1}_{j_*F}(n)$ are defined as the colimit arrow of $F(f) : \mathbf{Set}(F(j(m)), F(j(n)))$ for all $f : \mathbf{R}(j(m), j(n))$, which concludes the proof. □

Lemma 6.4. *The embeddings of* **R** *into* $\mathbf{Pshf}_L$ *and* $P\mathbf{R}$ *commute with the equivalence between these categories.*

Proof. Note:

$$(ﻉ \bullet j_*)(A)(n) = (j \bullet (ﻉ(A)))(n) = ﻉ(A)(j(n)) = \mathbf{R}(U^n, A)$$

$$(\iota\bullet \sim)(A)(n) = \{a : L_n \mid \iota_{0,n}(A)a = a\}$$

By Scott's representation theorem, we have $\varphi_n : \mathbf{R}(U^n, U) \xrightarrow{\sim} L_n$, given by

$$\varphi_n(f) = \iota_{0,n}(f)(x_i)_i.$$

Restricting this equivalence to the morphisms to A yields

$$\mathbf{R}(U^n, A) \cong \{a : L_n \mid Aa = a\}.$$

Of course, this is natural in A, since for $g : \mathbf{R}(A, B)$, postcomposing elements of $\mathbf{R}(U^n, A)$ by g is equivalent to applying g to the elements of $\{a : L_n \mid Aa = a\}$.

Lastly, for $f : L_m^n$, the presheaf actions

$$(ﻉ \bullet j_*)(A)(f) : \mathbf{R}(U^n, A) \to \mathbf{R}(U^m, A)$$

and

$$(\iota\bullet \sim)(A)(f) : \{a : L_n \mid \iota_{0,n}(A)a = a\} \to \{a : L_m \mid \iota_{0,n}(A)a = a\}$$

are in fact given by the substitution operations $\bullet$ in $E(U)$ and L. Since Scott's representation theorem shows that $E(U)$ is isomorphic to L as a λ-theory, these substitutions are compatible with the φ_n, which shows naturality in n. □

6.3 Relative Cartesian Closedness of the Category of Retracts

Recall that in the last chapter, we saw that Paul Taylor shows that the category of retracts **R**, studied by Scott, is not only cartesian closed, but also 'relatively cartesian closed'. To make his proof work in univalent foundations, we need the axiom of choice. The category in question is the Karoubi envelope of the category C_{L_0} associated to the monoid $(L_0, \circ)$.

Now, recall from Chapter 2, that in univalent foundations there are three different definitions that we can take for the Karoubi envelope, denoted $\overline{C}_{L_0}$, $\tilde{C}_{L_0}$ and $\hat{C}_{L_0}$. There is three of them because in classical category theory there are two definitions that are equivalent, and because for one of the definitions we have to choose whether we interpret the 'existence' of a retraction as additional structure (the retraction is given explicitly) or as a property (where we demand 'mere existence'). This gives the following diagram (Corollary 2.75):

$$C \xhookrightarrow{\iota_C} \overline{C} \xrightarrow{\sim} \tilde{C} \hookrightarrow\!\!\!\!\rightarrow \hat{C} \quad \subseteq \quad PC$$

where $\overline{C}_{L_0}$ and $\tilde{C}_{L_0}$ are equivalent to each other, but usually not to $\hat{C}_{L_0}$. We saw that $\hat{C}_{L_0}$ is the Rezk completion of $\overline{C}_{L_0}$ and $\tilde{C}_{L_0}$.

This is a fine example of a situation where notions that are equivalent in classical mathematics actually diverge in univalent foundations. We might say that this is annoying, because we have to be more careful which definition we pick. On the other hand, this gives 'more detail' in some sense: it is interesting to study which proofs work in which context, and this tells us something about the nature of the proofs.

The proofs of Scott and Taylor are about $\overline{C}_{L_0}$, because they work explicitly with idempotents. The question whether it is possible to lift the proofs to $\tilde{C}_{L_0}$ or $\hat{C}_{L_0}$, is left for future research. On the other hand, as we will see, Hyland works with presheaves and in this way, he shows that $\hat{C}_{L_0}$ is relatively cartesian closed.

Recall that in this thesis, we use the notation Δ_X for the functor $C \to (C \downarrow X)$, sending Y to $p_1 : X \times Y \to X$. Also, we use the notation $X \lhd Y$ to denote the type of retraction-section

pairs $r : Y \to X$ and $s : X \to Y$. We will sometimes not name $(r, s) : X \lhd Y$ explicitly, but just mention that we have $X \lhd Y$, meaning that we have such r and s.

Now, officially, $\hat{C}_{L_0}$ is the full subcategory of PC_{L_0}, consisting of all objects $X : PC_{L_0}$ such that there merely exists $X \lhd \text{ℶ}(\star)$. However, recall from the previous section that $PC_{L_0} \cong PC_{L_1} \cong \mathbf{Pshf}_L$. Under this equivalence, $\text{ℶ}(\star)$ is embedded as the theory presheaf L, and we will work in the full subcategory $\hat{\mathbf{R}} \subseteq \mathbf{Pshf}_L$ of the objects $X : \mathbf{Pshf}_L$ such that there merely exists $X \lhd L$. Of course, then $\hat{\mathbf{R}}$ is equivalent to $\hat{C}_{L_0}$.

Remark 6.5. There are a couple of tricks that we will use in this section. First of all, note that retractions can be composed. That is, given $(r, s) : X \lhd Y$ and $(r', s') : Y \lhd Z$, we have

$$(r' \cdot r, s \cdot s') : X \lhd Z.$$

Therefore, if we need to show that X is a retract of Z, and we know that X is a retract of Y, it remains to show that Y is a retract of Z.

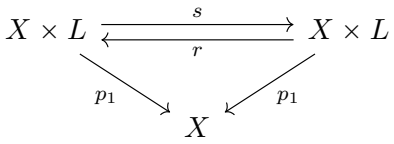
Also, functors preserve retractions, so if we have a functor $F : C \to D$ and we have $(r, s) : X \lhd Y$, then we have

$$(F(r), F(s)) : F(X) \lhd F(Y).$$

Lastly, if we want to show that we can do some construction on some $X : \hat{\mathbf{R}}$, we can often borrow the construction from $\mathbf{Pshf}_L$ to get some object $Y : \mathbf{Pshf}_L$. For this part, we cannot use the $(r, s) : X \lhd L$. To show that Y is indeed in $\hat{\mathbf{R}}$, we must show that there merely exist $(r', s') : Y \lhd L$. By the recursion principle of the propositional truncation, we can assume that we have a concrete $(r, s) : X \lhd L$ to construct r' and s'.

Recall that Paul Taylor's display maps for $\mathbf{R}$ can be characterized as the retracts of $\Delta_X U$ (the projection $p_1 : X \times U \to X$) in $(\mathbf{R} \downarrow X)$. This idea also works for $\hat{\mathbf{R}}$, and Hyland defines $D(Y, X) \subseteq \hat{\mathbf{R}}(Y, X)$ to consist of the retracts of $\Delta_X L$ in $(\hat{\mathbf{R}} \downarrow X)$. Note that, because $\hat{\mathbf{R}}$ is a full subcategory of $\mathbf{Pshf}_L$, $(\hat{\mathbf{R}} \downarrow X)$ is equivalent to $(\mathbf{Pshf}_L \downarrow X)$ for $X : \hat{\mathbf{R}}$.

Remark 6.6. Again, being a retract of $\Delta_X L$ is usually not a mere proposition. For example, consider the ways in which $\Delta_X L$ is a retract of itself:



In this particular case, the fact that r and s are morphisms in the slice category requires that $r \cdot p_1 = p_1$ and $s \cdot p_1 = p_1$, but then we can take

$$r = p_1 \times f \quad \text{and} \quad s = p_1 \times g$$

(or more generally $\langle p_1, f' \rangle$ and $\langle p_1, g' \rangle$) for any f and g that satisfy $g \cdot f = \mathrm{id}_L$. Two options are

$$f = g = \mathrm{id}_L \quad \text{or} \quad f_n(t) = t(\lambda x_{n+1}, x_{n+1}), \quad g_n(t) = \lambda x_{n+1}, \iota_{n,1}(t),$$

and these options are distinct if L is nontrivial.

Therefore, if we want $D(Y, X)$ to be a subset of $\hat{\mathbf{R}}(Y, X)$, we need to take the propositional truncation of the existence of r and s.

Remark 6.7. Recall that for all $X : \mathbf{R}$, we have $X \lhd U = \lambda x_1, x_1$, so by the above, for all Y in the image of the embedding of $\mathbf{R}$ into $\mathbf{Pshf}_L$, we have $Y \lhd L$. In particular, Y is in $\hat{\mathbf{R}}$. Actually, we already knew this, because this essentially describes the embedding of $\mathbf{R}$ into its Rezk completion $\hat{\mathbf{R}}$.

An important example is $U \times U : \mathbf{R}$, which maps to something isomorphic to $L \times L$ (because the Yoneda embedding and any adjoint equivalence preserves limits). The retraction $(r, s) : L \times L \lhd L$ is given explicitly by the (pairing and splitting) morphisms

$$s_n(a, b) = \lambda x_{n+1}, x_{n+1}ab \quad \text{and} \quad r_n(a) = (a(\lambda x_{n+1}x_{n+2}, x_{n+1}), a(\lambda x_{n+1}x_{n+2}, x_{n+2})).$$

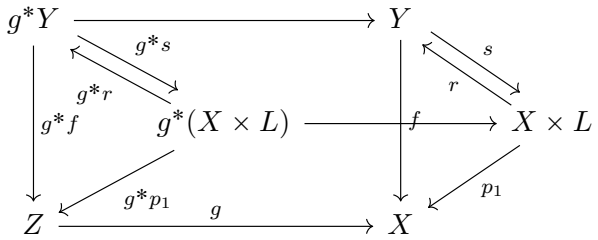
Now, for any $X, Y : \mathbf{Pshf}_L$ and $f : \mathbf{Pshf}_L(Y, X)$, if we have $(Y, f) \lhd \Delta_X L$ in $\mathbf{Pshf}_L$, we have in particular $Y \lhd X \times L$. If we also have $X \lhd L$, we also have $X \times L \lhd L \times L$ because $- \times L$ is a functor. Then we have

$$Y \lhd X \times L \lhd L \times L \lhd L.$$

In particular, for any $X : \hat{\mathbf{R}}$, if some $(Y, f) : (\mathbf{Pshf}_L \downarrow X)$ is a retract of $\Delta_X L$, we know that Y is in $\hat{\mathbf{R}}$.

Note that D is indeed a class of display maps:

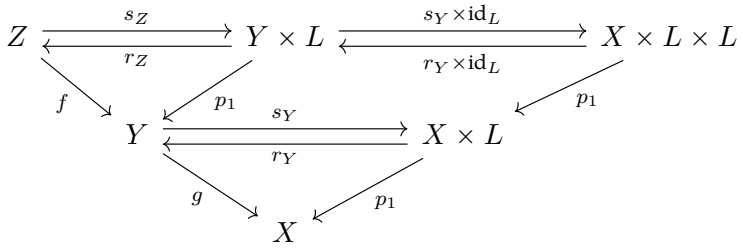
1. Take $f : D(Y, X)$ and $g : \hat{\mathbf{R}}(Z, X)$. We can borrow the pullback from $\mathbf{Pshf}_L$ to get $g^*(Y, f) : (\mathbf{Pshf}_L \downarrow Z)$. We now must show that there merely exists $g^*(Y, f) \lhd \Delta_Z L$ and in this part, we can assume that the mere existences of retractions are actually given by concrete retraction-section pairs.
   Now, note that taking pullbacks gives a functor $g^* : (\mathbf{Pshf}_L \downarrow X) \to (\mathbf{Pshf}_L \downarrow Z)$. Now, $(Y, f) \lhd \Delta_X L$, gives $(g^*Y, g^*f) \lhd g^*(\Delta_X L)$:



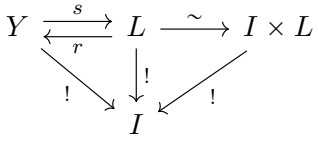
   The isomorphism between $g^*(\Delta_X L)$ and $\Delta_Z L$ (Remark 2.34) shows that we have $(g^*Y, g^*f) \lhd \Delta_Z L$. From this, it also follows that g^*Y is in $\hat{\mathbf{R}}$.
   Now, g^*Y is the pullback of f and g in $\mathbf{Pshf}_L$, and $\hat{\mathbf{R}}$ is a full subcategory of $\mathbf{Pshf}_L$, so g^*Y is also the pullback of f and g in $\hat{\mathbf{R}}$.
2. Take $f : D(Z, Y)$ and $g : D(Y, X)$. We have to show that there merely exists $(Z, f \cdot g) \lhd \Delta_X L$. Because this is a proposition, we can assume that we have

$$(r_Z, s_Z) : (Z, f) \lhd \Delta_Y L \quad \text{and} \quad (r_Y, s_Y) : (Y, g) \lhd \Delta_X L.$$

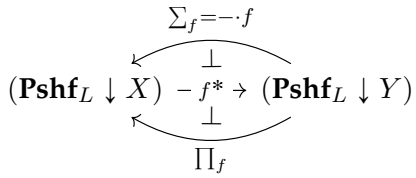
   Then the following diagram gives $(Z, f \cdot g) \lhd \Delta_X(L \times L)$:



   Now, as we saw earlier, Δ_X is a functor and $L \times L$ is a retract of L. Therefore, we have $\Delta_X(L \times L) \lhd \Delta_X L$ and $f \cdot g$ is in $D(Z, X)$.
3. Note that $\mathbf{Pshf}_L$ has a terminal presheaf $I_n = \{\star\}$. Under the embedding of $\mathbf{R}$ into $\mathbf{Pshf}_L$, this is (isomorphic to) the image of $\lambda x_1 x_2, x_2 : \mathbf{R}$, and therefore, it is in $\hat{\mathbf{R}}$. Now, given $Y : \hat{\mathbf{R}}$, we need to show that the terminal projection ! is in $D(Y, I)$. Note that any morphism trivially commutes with the terminal projections. Since $Y : \hat{\mathbf{R}}$, we have $(r, s) : Y \lhd L$, and since $L \cong I \times L$, this means that the terminal projection is indeed in $D(Y, I)$:



Recall that $\mathbf{Pshf}_L$ is isomorphic to the presheaf category $P\mathbf{L}$, so it is an elementary topos and it is locally cartesian closed (Example 5.2.5 and Theorem 5.8.4, [Bor94], Volume 3). Therefore, for a morphism $f : \mathbf{Pshf}_L(Y, X)$, we have adjunctions

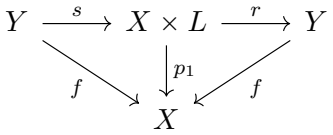


Now, Hyland shows the following:

Theorem 6.8. *For $f : D(Y, X)$, we can restrict $\sum_f$ and $\prod_f$ to functors from $(\hat{\mathbf{R}} \downarrow_D Y)$ to $(\hat{\mathbf{R}} \downarrow_D X)$.*

Proof. Given $(Z, g) : (\mathbf{Pshf}_L \downarrow Y)$, for $\prod_F$ we only have to show that if we have $(Z, g) \lhd \Delta_Y L$, then we have $\prod_f (Z, g) \lhd \Delta_X L$. Note that since X is in $\hat{\mathbf{R}}$, this also shows that $\prod_f Z$ is in $\hat{\mathbf{R}}$. Because functors preserve retractions, it suffices to show that we have $\prod_f \Delta_Y L \lhd \Delta_X L$.

Since $f : D(Y, X)$, we have a retraction



The counits of the adjunctions $r_* \vdash \prod_r$ and $s_* \vdash \prod_s$, give maps

$$\prod_f \Delta_Y L \xrightarrow{\prod_f \eta_r} \prod_f \prod_r r^* \Delta_Y L \xrightarrow{\prod_f \prod_r \eta_s} \prod_f \prod_r \prod_s s^* r^* \Delta_Y L$$

Their composite is $\prod_f \eta_{s \cdot r}$. However, note that $r \cdot s = \mathrm{id}_Y$, so we have an isomorphism of functors

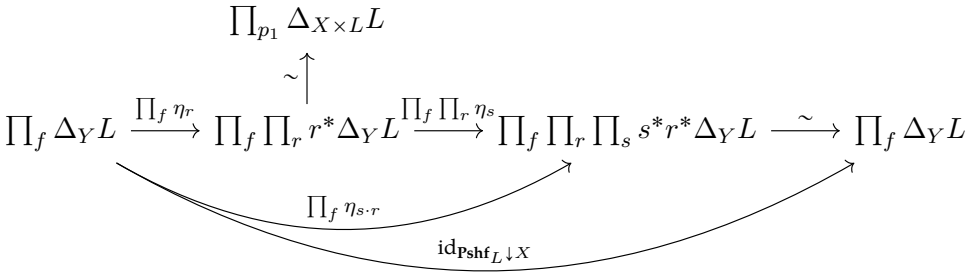
$$(s \cdot r)^* \cong \mathrm{id}_{\mathbf{Pshf}_L \downarrow Y} \quad \text{and} \quad \prod_{s \cdot r} \cong \mathrm{id}_{\mathbf{Pshf}_L \downarrow Y}$$

and transporting $\eta_{s \cdot r}$ along these isomorphisms, we get the identity natural transformation $\mathrm{id}_{\mathbf{Pshf}_L \downarrow Y} \Rightarrow \mathrm{id}_{\mathbf{Pshf}_L \downarrow Y}$.

Secondly, note that $r \cdot f = p_1$, so we have $\prod_r \bullet \prod_f \cong \prod_{p_1}$.

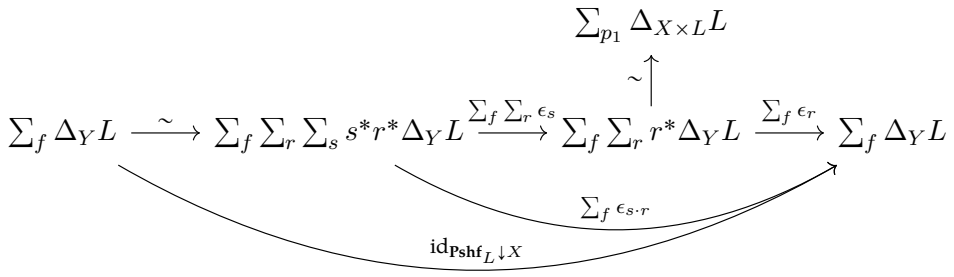
Lastly, by Remark 2.34, $r^* \Delta_Y L \cong \Delta_{X \times L} L$.

Therefore, we have the following diagram, showing that $\prod_f \Delta_Y L$ is a retract of $\prod_{p_1} \Delta_{X \times L} L$:



So it suffices to show that $\prod_{p_1} \Delta_{X\times L} L$ is a retract of $\Delta_X L$. By Lemma 2.35, we have $\prod_{p_1} \Delta_{X\times L} L \cong \Delta_X L^L$. By functoriality of Δ_X, it suffices to show that L^L is a retract of L. But we already saw in Theorem 6.1 that $L^L \cong A(L,1)$ is a retract of L. This shows that $\prod_f$ indeed restricts to a functor from $(\hat{\mathbf{R}} \downarrow_D Y)$ to $(\hat{\mathbf{R}} \downarrow_D X)$.

The proof for $\sum_f$ is very similar. $\sum_f \Delta_Y L$ is a retract of $\sum_{p_1} \Delta_{X\times L} L$:



By Lemma 2.31, the functor $\sum_{p_1}$ is given by postcomposition, so $\sum_{p_1} \Delta_{X\times L} L \cong \Delta_X (L \times L)$. As we saw earlier, we have $L \lhd L \times L$, which completes the proof that $\sum_f$ restricts to a functor from $(\hat{\mathbf{R}} \downarrow_D Y)$ to $(\hat{\mathbf{R}} \downarrow_D X)$. □

Corollary 6.9. *Since the $(\hat{\mathbf{R}} \downarrow_D X)$ are full subcategories of the $(\mathbf{Pshf}_L \downarrow X)$, so we can restrict the adjunction $f^* \dashv \prod_f$ to an adjunction $\bar{f}^* \dashv \bar{\prod}_f$ between $(\hat{\mathbf{R}} \downarrow_D X)$ and $(\hat{\mathbf{R}} \downarrow_D Y)$, which shows that $\hat{\mathbf{R}}$ is cartesian closed relative to the class D of display maps.*

Remark 6.10. It is a nice feature of Hyland's approach that in univalent foundations, the proofs that D is a class of display maps and that we can restrict $\prod_f$ and $\sum_f$ work without the axiom of choice.

Remark 6.11. Recall that to make sure that $D(Y,X)$ is a subset of $\hat{\mathbf{R}}(Y,X)$, we had no choice but to take the propositional truncation of the $(Y,f) \lhd \Delta_X L$. Coincidentally, for $\hat{\mathbf{R}}$ we also took the propositional truncation of the $X \lhd L$.

In the proofs that D is a class of display maps, and that we can restrict $\prod_f$, this worked out great. A couple of times, we first did a construction on the presheaves, ignoring the mere existence of the retractions, and then we showed mere existence of one or two retractions, for which we were allowed to assume that we actually had concrete retractions from L or $\Delta_X L$.

Therefore, this only works in $\hat{\mathbf{R}} \cong \hat{C}_{L_0}$. Suppose that we tried to do this in $\tilde{C}_{L_0}$. Note that this category is equivalent to the category with as objects $X : \mathbf{Pshf}_L$, together with some $(r,s) : X \lhd L$. Trying to define the pullback functor would already cause problems. To define a pullback, we would need to explicitly construct some retraction $g^*Y \lhd L$, for which we would need a retraction $Z \lhd L$. However, we have only mere existence of this retraction, so we would only be able to show mere existence of pullbacks, like in Taylor's proof. In Taylor's case, we were able to use the axiom of choice because the objects of $\mathbf{R}$ form a set. However, the objects of $\tilde{\mathbf{R}}$ do not form a set, and so we cannot use the axiom of choice here (Lemma 3.8.5 in [Uni13] shows why the axiom of choice only works for sets).

Since we have an adjoint equivalence between $\tilde{C}_{L_0}$ and $\overline{C}_{L_0} = \mathbf{R}$, and since any reasoning about $\mathbf{R}$ using presheaves would use this equivalence, we would run into the same problems if we tried try to apply Hyland's approach to $\mathbf{R}$. However, note that there we might be able to use the axiom of choice to show mere existence of a pullback functor, like in Taylor's proof.

6.4 An Elementary Proof of the 'Fundamental Theorem'

The final theorem that Hyland works towards in his paper constructs an equivalence between two ways to reason about the λ-calculus using these tools from universal algebra: λ-theories

and Λ-algebras. In this section, we will provide an elementary proof of this theorem. Then, in the next section, we will give Hyland's original proof, which uses more high-level categorical reasoning.

We will assume that Λ (and therefore, any λ-theory) satisfies β-equality.

In both cases, one part of the equivalence is very easy. Recall that the pure λ-calculus, the λ-theory Λ, is the initial object of **LamTh**. That means that **LamTh** is equivalent to $(\Lambda \downarrow \mathbf{LamTh})$. Now, using Definition 4.55, for any n we get a functor

$$-_n : \mathbf{LamTh} \to \mathbf{Alg}_\Lambda.$$

Here we are only interested in the case $n = 0$.

We will use Lemma 3.4, by showing that $-_0$ is a weak equivalence.

First of all, let A be an algebra for the initial λ-theory Λ.

The Λ-algebra structure gives the terms of A a lot of interesting behaviour. For example, we can define 'function application' and composition as

$$ab = (x_1x_2) \bullet (a, b) \quad \text{and} \quad a \circ b = (x_1 \circ x_2) \bullet (a, b),$$

and the same for the other constructions at the start of Section 5.2.

Remark 6.12. Recall that in Example 4.46, we constructed an algebraic theory T encapsulating the structure of a monoid. This allowed us to define a monoid operation on T-algebras as well. We then were able to transfer associativity of the operation on the T_n to associativity of the operation on the algebras. In exactly the same way, the function composition on A is associative because composition on Λ_n is associative. Similarly, we can show that $\pi_1(a, b) = a$, that $\pi_2 \circ \langle a, b\rangle = (\lambda x_2, x_1x_2) \bullet b$ etc.

In fact, we can repeat almost the entirety of Section 5.2 for A instead of for L_0:

Definition 6.13. We define the 'category of retracts' of A as the category $\mathbf{R}_A$ given by

$$(\mathbf{R}_A)_0 = \{X : A \mid X \circ X = X\} \quad \text{and} \quad \mathbf{R}_A(X, Y) = \{f : A \mid Y \circ f \circ X = f\}.$$

Just like in Section 5.2, $\mathbf{R}_A$ has 'universal object' and terminal object

$$U = (\lambda x_1, x_1) \bullet () \quad \text{and} \quad I = (\lambda x_1, c_1),$$

products

$$A \times B = \langle A \circ \pi_1, B \circ \pi_2\rangle$$

and exponential objects

$$B^A = (\lambda x_3, x_1 \circ x_3 \circ x_2) \bullet (B, A)$$

with the isomorphism $\psi : \mathbf{R}_A(C \times A, B) \xrightarrow{\sim} \mathbf{R}_A(C, B^A)$ given by

$$\psi(f) = (\lambda x_2x_3, x_1(x_2, x_3)) \bullet f \quad \text{and} \quad \psi^{-1}(f) = (\lambda x_2, x_1(\pi_1x_2)(\pi_2x_2)) \bullet f.$$

Therefore, we have a λ-theory $E_{\mathbf{R}_A}(U)$.

Remark 6.14. Note that if $A = L_0$, then the construction of $\mathbf{R}_A$ and $\mathbf{R}$ coincide, so we have an almost trivial equivalence $\mathbf{R}_A \cong \mathbf{R}$, and an isomorphism $E_{\mathbf{R}}(U) \cong E_{\mathbf{R}_A}(U)$.

The following lemma shows that our functor is essentially surjective:

Lemma 6.15. *We have a Λ-algebra isomorphism $\epsilon_A : E_{\mathbf{R}_A}(U)_0 \xrightarrow{\sim} A$.*

Proof. Take $c_n = (\lambda x_{n+1}, x_{n+1}) : \Lambda_n$. Note that

$$E(U)_0 = \{f : A \mid (\lambda x_2, x_1 c_2) \bullet f = f\}.$$

Therefore, we have a bijection given by

$$\epsilon(a) = (x_1 c_1) \bullet a \quad \text{and} \quad \epsilon^{-1}(a) = (\lambda x_2, x_1) \bullet a.$$

Now, to show that this is an isomorphism of Λ-algebras, recall that $\bullet : \mathbf{R}_A(U^n, U) \times \mathbf{R}_A(I, U)^n \to \mathbf{R}_A(U)$ is given by precomposition with the product morphism and that the Λ-algebra structure on $E(U)_0$ is given by the embedding of Λ into $E(U)_0$, given by $f \mapsto \iota_{\Lambda n}(f) \bullet_A ()$ for some collection $(\iota_{\Lambda n} : \Lambda_n \to \Lambda_0)_n$ (from now on, we will drop the n and just write ι_Λ). We have for all $f : \Lambda_n$, and all $a : E(U)_0^n$,

$$\begin{aligned}
\epsilon(f \bullet_{E(U)} a) &= ((\iota_\Lambda(f) \bullet_A ()) \circ \langle a_i \rangle_i)(c_n) \\
&= (\iota_{0,n}(\iota_\Lambda(f))(x_i c_n)_i) \bullet_A a \\
&= (\iota_{0,n}(\iota_\Lambda(f))(x_i)_i) \bullet_A (\epsilon(a_i))_i,
\end{aligned}$$

and we want this to equal $f \bullet_A (\epsilon(a_i))_i$. Leaving out the $\iota_{0,n}$, all we need to do is show that for all $f : \Lambda_n$, we have $\iota_\Lambda(f)(x_i)_i = f$. We will do this by structural induction on f.

- If $f = x_{n,i}$, we have $\iota_\Lambda(f) = \pi_{n,i}$, and
$$\iota_\Lambda(f)(x_i)_i = \pi_{n,i}(x_{n,i})_i = x_{n,i}.$$
- If $f = \lambda x_{n+1}, g$ for some $g : \Lambda_{n+1}$, we have
$$\iota_\Lambda(f) = (\lambda x_1 x_2, \iota_{0,2}(\iota_\Lambda(g))(x_1, x_2)),$$
and if the induction hypothesis holds for g, then
$$\begin{aligned}
\iota_\Lambda(f)(x_i)_i &= (\lambda x_{n+1} x_{n+2}, \iota_\Lambda(g)(x_{n+1}, x_{n+2}))(x_i)_i \\
&= \lambda x_{n+1}, \iota_\Lambda(g)((x_i)_i, x_{n+1}) \\
&= \lambda x_{n+1}, g.
\end{aligned}$$
- Recall that application in a λ-theory is given by $g_1 g_2 = \rho(g_1) \bullet (x_1, \ldots, x_n, g_2)$. Now, if $f = g_1 g_2$ for $g_1, g_2 : \Lambda_{n+1}$, we have
$$\iota_\Lambda(f) = (\lambda x_1, \iota_\Lambda(g_1)(\pi_1 x_1)(\pi_2 x_1)) \circ \langle \mathrm{id}_{U^n}, \iota_\Lambda(g_2) \rangle = \lambda x_1, \iota_\Lambda(g_1) x_1 (\iota_\Lambda(g_2) x_1),$$
and if the induction hypothesis holds for g_1 and g_2, then
$$\begin{aligned}
\iota_\Lambda(f)(x_i)_i &= (\lambda x_{n+1}, \iota_\Lambda(g_1) x_{n+1} (\iota_\Lambda(g_2) x_{n+1}))(x_i)_i \\
&= \iota_\Lambda(g_1)(x_i)_i (\iota_\Lambda(g_2)(x_i)_i) \\
&= g_1 g_2.
\end{aligned}$$

□

Now, take a morphism $F : \mathbf{Alg}_\Lambda(A, B)$. Recall that $\circ$, the categories $\mathbf{R}_A$ and $\mathbf{R}_B$, and their products and exponential objects are given using $f \bullet (a_2, \ldots, a_n)$ for some $f : \Lambda_n$ and $a_1, \ldots, a_n : A$. Since we have $F(f \bullet (a_i)_i) = f \bullet (F(a_i))$, F gives a functor $F : \mathbf{R}_A \to \mathbf{R}_B$, such that

$$F(I) = I, F(U) = U, F(A \times B) = F(A) \times F(B) \quad \text{and} \quad F(A^B) = F(A)^{F(B)},$$

and the same for the product projections and the the natural isomorphisms $\mathbf{R}_A(C, B^A) \cong \mathbf{R}_A(C \times A, B)$. In particular, we have for all $f : \mathbf{R}_A(U^n, U)$,

$$F(f) : \mathbf{R}_B(F(U^n), F(U)) = \mathbf{R}_B(U^n, U).$$

Therefore, F gives a Λ-theory morphism between the endomorphism theories of $U : \mathbf{R}_A$ and $U : \mathbf{R}_B$. This allows us to show:

Lemma 6.16. *The functor is full.*

Proof. For any λ-theory L, Theorem 5.8 gives an isomorphism of λ-theories $\eta_L : L \xrightarrow{\sim} E_{\mathbf{R}_{L_0}}(U)$, with

$$\eta_{L0}(f) = \lambda x_1, \iota_{0,1}(f) \quad \text{and} \quad {\eta_L^{-1}}_0(g) = g(\lambda x_1, x_1).$$

Now, for $L, L' : \mathbf{LamTh}$ and $F : \mathbf{Alg}_\Lambda(L_0, L'_0)$, we have

$$\eta_L \cdot F \cdot \eta_{L'}^{-1} : \mathbf{LamTh}(L, L'),$$

and we have for all $s : L_0$,

$$(\eta_{L0} \cdot F \cdot {\eta_{L'}^{-1}}_0)(s) = (\lambda x_1, \iota_{0,1}(F(s)))(\lambda x_1, x_1) = F(s),$$

which shows that the functor is full. □

Now, we only have to show:

Lemma 6.17. *The functor is faithful.*

Proof. Take morphisms $F, G : \mathbf{LamTh}(L, L')$. Suppose that $F_0 = G_0$. Then, for all $s : L_n$, we have

$$F_n(s) = \rho^n(F_0(\lambda^n(s))) = \rho^n(G_0(\lambda^n(s))) = G_n(s),$$

so $F = G$. □

Summarizing,

Theorem 6.18. *The functor that sends $L : \mathbf{LamTh}$ to $L_0 : \mathbf{Alg}_\Lambda$, is an adjoint equivalence.*

Proof. By Lemma 6.16 and Lemma 6.17, the functor is fully faithful. By Lemma 6.15, it is essentially surjective, so it is a weak equivalence. Since **LamTh** and $\mathbf{Alg}_\Lambda$ are univalent categories, Lemma 3.4 shows that F is an adjoint equivalence. □

6.5 Hyland's Proof

Hyland gives a more category theoretical proof. That means that there is more high-level intuition why things work the way they work, but on the flip side, there is a lot more details to check.

6.5.1 Terms of a Λ-algebra

Definition 6.19. Take $\mathbf{1}_n = (\lambda x_1 \dots x_n, x_1 \dots x_n) \bullet () : A$.

Remark 6.20. Note that we have $\mathbf{1}_1 = U : \mathbf{R}_A$.

Definition 6.21. In this section, we consider sets of elements of A that behave like functions in n variables:

$$A_n = \{a : A \mid (\lambda x_2 x_3 \dots x_{n+1}, x_1 x_2 x_3 \dots x_{n+1}) \bullet a = a\}.$$

Remark 6.22. Some straightforward rewriting, shows that

$$A_n = \{a : A \mid \mathbf{1}_n \circ a = a\}.$$

Remark 6.23. Also note that $\mathbf{1}_n \circ a \circ \mathbf{1}_n = \mathbf{1}_n \circ a$, so for $a : A_n$, $a \circ \mathbf{1}_n = a$.

The following shows that 'functions in n variables' are indeed all in A_n:

Lemma 6.24. *For $t : \Lambda_{m+n}$ and $a_1, \ldots, a_m : A$, we have $(\lambda^n t) \bullet (a_1, \ldots, a_m) : A$ and we have*

$$\mathbf{1}_n \circ ((\lambda^n t) \bullet (a_1, \ldots, a_m)) = (\lambda^n t) \bullet (a_1, \ldots, a_m),$$

so $(\lambda^n t) \bullet (a_1, \ldots, a_m) : A_n$.

Proof. This follows by straightforward rewriting. □

Corollary 6.25. *By the previous remark,*

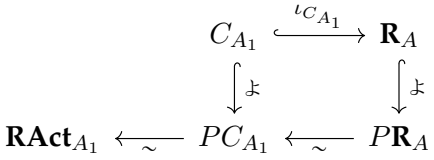
$$((\lambda^n t) \bullet (a_1, \ldots, a_m)) \circ \mathbf{1}_n = (\lambda^n t) \bullet (a_1, \ldots, a_m).$$

Corollary 6.26. *In particular, $\mathbf{1}_m \circ \mathbf{1}_n = \mathbf{1}_{\max(m,n)}$. From this, it follows that $A_m \subseteq A_n$ for $m \leqslant n$. It also follows that $a \mapsto \mathbf{1}_n \circ a$ gives a function from A to A_n (and also from $A_m \subseteq A$ to A_n).*

Definition 6.27 (`lambda_algebra_monoid`)**.** We make A_1, the 'functional elements' of A, into a monoid under composition $\circ$ with unit $\mathbf{1}_1$. The fact that this is a monoid follows from the remarks above.

Recall that we have an equivalence $[C_{A_1}^{\mathrm{op}}, \mathbf{Set}] \cong \mathbf{RAct}_{A_1}$.

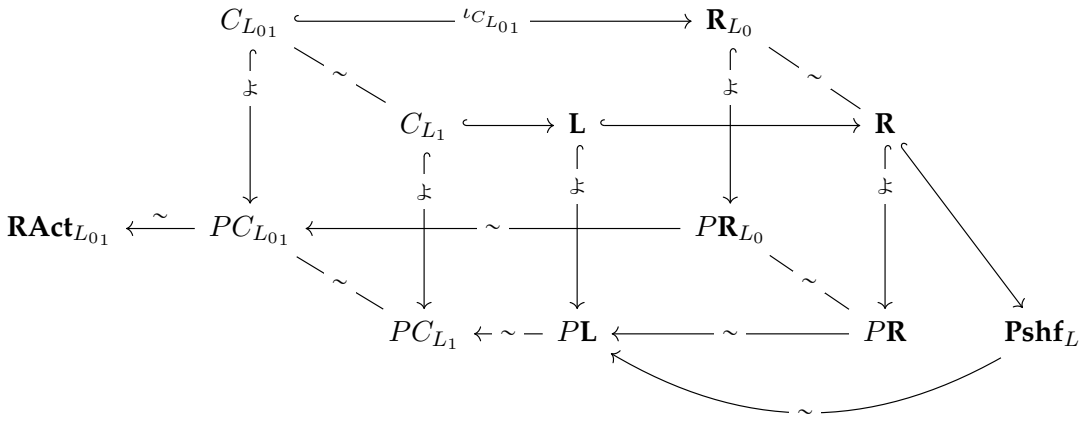
Remark 6.28. Note that, like in the last chapter, $\mathbf{R}_A$ pops up as the Karoubi envelope of the monoid category C_{A_1}, and fits into the following diagram:



Explicitly, this gives the embedding $\mathbf{R}_A(\mathbf{1}_1, -) : \mathbf{R}_A \hookrightarrow \mathbf{RAct}_{A_1}$ given by

$$X \mapsto \mathbf{R}_A(\mathbf{1}_1, X) = \{x : A \mid X \circ x = x\}.$$

Also, note that if $A = L_0$ for some λ-theory L, then the monoid A_1 is equivalent to the monoid L_1, and $\mathbf{R}_A$ is equivalent to our familiar category of retracts $\mathbf{R}$. Using Lemma 6.3, we have the following 2-commutative diagram:



6.5.2 Constructing a theory from an algebra

Definition 6.29. Composition $\circ$ gives a right A_1-action on the A_n, so we have $A_n : \mathbf{RAct}_{A_1}$. In particular, A_1 acts on itself, and we will call this set with right A_1-action U_A.

Lemma 6.30 (`universal_monoid_exponential_iso`)**.** *A_2 is isomorphic to the exponential object $U_A^{U_A}$ in $\mathbf{RAct}_{A_1}$.*

Proof. Recall that $U_A^{U_A}$ is the set of A_1-equivariant morphisms $U_A \times U_A \to U_A$. We have an isomorphism $\psi : A_2 \xrightarrow{\sim} U_A^{U_A}$, given by

$$\psi(f) = (b, b') \mapsto (\lambda x_4, x_1(x_2x_4)(x_3x_4)) \bullet (f, b, b'),$$

treating $f : A_2$ as a function in two variables, and simultaneously composing it with the functions $b, b' : A_1$. It has an inverse

$$\psi^{-1}(g) = (\lambda x_2 x_3, x_1(x_2, x_3)) \bullet (g(\pi_1, \pi_2)).$$

Note that the first pair (x_2, x_3) is a λ-term, whereas the second pair (π_1, π_2) is a pair in $U_A \times U_A$.

For $f : U_A^{U_A}$ and $(a_1, a_2) : U_A \times U_A$, we have

$$\psi(\psi^{-1}(f))(a_1, a_2) = f(\pi_1, \pi_2) \circ \langle a_1, a_2 \rangle = f(\pi_1 \circ \langle a_1, a_2 \rangle, \pi_2 \circ \langle a_1, a_2 \rangle) = f(a_1, a_2).$$

Here we use the A_1-equivariance of f. In the last step of this proof, we use, among other things, the fact that the $a_i : A_1$ and therefore $\lambda x_1, a_i x_1 = a_i$.

Some straightforward rewriting shows that for $a : A_2$, we have $\psi^{-1}(\psi(a)) = a$. In the last step of this proof, we use the fact that $a : A_2$ and therefore $\lambda x_1 x_2, a x_1 x_2 = a$.

Therefore, ψ is a bijection and, as it turns out, an isomorphism. □

Remark 6.31. Recall that the embedding $\mathbf{R}_A \hookrightarrow \mathbf{RAct}_{A_1}$ is the composition of a Yoneda embedding and two adjoint equivalences. These all preserve exponential objects (see Lemma 2.19). Now, note that A_1 is the image of $U : \mathbf{R}_A$, so the exponential $U_A^{U_A}$ is A_2, the image of $U^U = \lambda x_1 x_2, x_1 x_2 = \mathbf{1}_2$.

Definition 6.32 (`lambda_algebra_theory`)**.** Recall that $A_2 \subseteq U_A$, and that we have a retraction $a \mapsto \mathbf{1}_2 \circ a : U_A \to A_2$.

Therefore, U_A is a reflexive object in $\mathbf{RAct}_{A_1}$, and we get a λ-theory $E(U_A)$.

Remark 6.33. The cartesian closed embedding of $\mathbf{R}_A$ into $\mathbf{RAct}_{A_1}$ sends U to U_A. It sends the retraction $U^U : \mathbf{R}_A(U, U^U)$ exactly to the retraction $a \mapsto \mathbf{1}_2 \circ a : \mathbf{RAct}_{A_1}(U_A, A_2)$. Therefore, we have a λ-theory isomorphism $E_{\mathbf{R}_A}(U) \cong E_{\mathbf{RAct}_{A_1}}(U_A)$.

6.5.3 Constructing a theory morphism from an algebra morphism

Take a morphism $F : \mathbf{Alg}_\Lambda(A, B)$. Note that F preserves $\circ$ and the $\mathbf{1}_n$. Therefore, it induces a monoid morphism $A_1 \to B_1$, which gives a functor $C_{A_1} \to C_{B_1}$. Precomposing with this, we get a functor $\mathbf{RAct}_{B_1} \to \mathbf{RAct}_{A_1}$.

To get a morphism $E_{\mathbf{RAct}_{A_1}}(U_A) \to E_{\mathbf{RAct}_{B_1}}(U_B)$ however, we need a functor $\mathbf{RAct}_{A_1} \to \mathbf{RAct}_{B_1}$, which we obtain by Left Kan extension: see Lemma 2.59 (the "tensor product").

Hyland gives a very high-level argument why F_* induces a morphism $\bar{F} : \mathbf{LamTh}(E_{\mathbf{RAct}_{A_1}}(U_A), E_{\mathbf{RAct}_{B_1}}(U_B$ which we will discuss a bit more here.

Lemma 6.34. *The extension of scalars F_* sends U_A to U_A and preserves finite products.*

Proof. By Lemma 2.60, we have $F_*(U_A) \cong U_A$.

We will show that F_* preserves binary products and the terminal object, because from this it follows that F_* preserves all finite products.

We use Lemma 2.61 to show that F_* preserves the terminal object. We take (very similar to the terminal object in $\mathbf{R}$):

$$a_0 = (\lambda x_1 x_2, x_2) \bullet () : A_1 \quad \text{and} \quad b_0 = (\lambda x_1 x_2, x_2) \bullet () : B_1$$

and a_0 is weakly terminal because for all $a : B_1$, we have $F(a_0) \circ a = b_0$.

We use Lemma 2.63 to show that F_* also preserves the product. Therefore, given $a_1, a_2 : B_1$. Take $b = \langle a_1, a_2 \rangle$, together with the familiar projections $\pi_i : A_1$. We have $a_i = F(\pi_i) \circ a$.

Now, for some $b' : B_1$ and $\pi_1', \pi_2' : A_1$ such that $a_i = F(\pi_i') \circ b'$, take $m = \langle \pi_1', \pi_2' \rangle$. Then $\pi_i \circ m = \pi_i'$ and $F(m) \circ a' = a$, so (a, π_1, π_2) is weakly terminal and F_* preserves binary products.

Since any finite product is (isomorphic to) a construction with a repeated binary product and the terminal object, the fact that F_* preserves binary products and the terminal object shows that F_* preserves all finite products. □

Now, we can start defining our lift of F:

Definition 6.35. We can send an element $g : E(U_A)_n = \mathbf{RAct}_A(U_A^n, U_A)$ to

$$F_*(g) : \mathbf{RAct}_{A'}(f(U_A^n), f(U_A)) \cong \mathbf{RAct}_{A'}((U_{A'})^n, U_{A'}) = E(U_{A'})_n$$

so we have a morphism $\bar{F} : \mathbf{LamTh}(E(U_A), E(U_{A'}))$.

Remark 6.36. The fact that $\bar{F}$ preserves the variables and substitution is not very hard, since these are just defined in terms of finite products of U_A and $\bar{F}$ preserves finite products and U_A.

However, showing that it is a λ-theory morphism is a different matter. Hyland claims

"F preserves $\mathbf{1}_n$, which determines the function space as a retract of the universal. So $\bar{F}$ preserves the retract and the result follows."

Although this covers the core of the argument, it is very complicated to actually verify that this works, because we need to pass through a lot of isomorphisms, and check that they work nicely together:

$$\begin{aligned} \alpha_A : \mathbf{RAct}_{A_1}(X \times Y, Z) &\xrightarrow{\sim} \mathbf{RAct}_{A_1}(X, Z^Y); \\ \beta : F_*(U_A) &\xrightarrow{\sim} U_{A'}; \\ \bar{\gamma} : F_*(A \times B) &\xrightarrow{\sim} F_*(A) \times F_*(B); \\ \gamma_n : F_*(X^n) &\xrightarrow{\sim} F_*(X)^n; \\ \delta_A : U_A^{U_A} &\xrightarrow{\sim} A_2. \end{aligned}$$

Since we already motivated this in a different way, we will leave it as an exercise for the enthousiastic reader.

Lemma 6.37. *For $F = id_A$, we have $\bar{F} = id_{E(U_A)}$.*

Proof. By the reasoning set forth in Lemma 6.17, we only need to check $\bar{F}_0$. Given $s : \mathbf{RAct}_{A_1}(I, U_A)$, we have

$$\bar{F}_0(s) = \gamma^{-1} \cdot F_*(s) \cdot \beta : \mathbf{RAct}_{A_1}(I, U_A)$$

for $\gamma : F_*(I) \cong I$ and $\beta : F_*(U_A) \cong U_A$. Then

$$\bar{\mathrm{id}}_A(s)(\star) = s(\star) \circ ((\lambda x_1 x_2, x_2) \bullet ()) = s(\star \circ ((\lambda x_1 x_2, x_2) \bullet ())),$$

so $\bar{\mathrm{id}}_{A0}(s) = s$ and we can conclude that $\bar{\mathrm{id}}_A = \mathrm{id}_{E(U_A)}$. □

Lemma 6.38. *For $F : \mathbf{Alg}_\Lambda(A, B)$ and $G : \mathbf{Alg}_\Lambda(B, C)$, we have $\bar{F} \cdot \bar{G} = \overline{F \cdot G}$.*

Proof. Again, we only have to check at the terms without free variables. Given $s : \mathbf{RAct}_{A_1}(I, U_A)$, we have

$$(\bar{F} \cdot \bar{G})(s) = \gamma_G^{-1} \cdot G_*(\gamma_F^{-1} \cdot F_*(s) \cdot \beta_F) \cdot \beta_G : \mathbf{RAct}_{C_1}(I, U_C)$$

for $\gamma : F_*(I) \cong I$ and $\beta : F_*(U_A) \cong U_A$. Then

$$\begin{aligned}(\bar{F} \cdot \bar{G})(s)(\star) &= G(F(s(\star)) \circ ((\lambda x_1 x_2, x_2) \bullet_B ())) \circ ((\lambda x_1 x_2, x_2) \bullet_C ()) \\ &= G(F(s(\star \circ ((\lambda x_1 x_2, x_2) \bullet_A ()) \circ ((\lambda x_1 x_2, x_2) \bullet_A ())))) \\ &= G(F(s(\star))) \\ &= (F \cdot G)^{-1}(s(\star \circ ((\lambda x_1 x_2, x_2) \bullet_A ()))) \\ &= \overline{(F \cdot G)}(s)(\star)\end{aligned}$$

so $(\bar{F} \cdot \bar{G})_0(s) = (\overline{F \cdot G})_0(s)$ and we can conclude that $\bar{F} \cdot \bar{G} = \overline{F \cdot G}$. □

Definition 6.39. We get a functor from $\mathbf{Alg}_\Lambda$ to **LamTh**, sending objects A to $E_{\mathbf{RAct}_{A_1}}(U_A)$ and morphisms $F : \mathbf{Alg}_\Lambda(A, B)$ to $\bar{F} : E(U_A) \to E(U_B)$.

Remark 6.40. Note that for $X : \mathbf{R}_A$, we have a natural isomorphism of sets with a right B_1-action

$$\psi : \{f : B \mid F(X) \circ f = f\} \xrightarrow{\sim} \{f : A \mid X \circ f = f\} \times B_1/\sim$$

given by

$$\psi(f) = (X, f) \quad \text{and} \quad \psi^{-1}(f, b) = F(f) \circ b.$$

Therefore, the following diagram 2-commutes:

$$\begin{array}{ccc} \mathbf{R}_A & \xrightarrow{F} & \mathbf{R}_B \\ \downarrow{\scriptstyle \mathbf{R}_A(\mathbf{1}_1, -)} & & \downarrow{\scriptstyle \mathbf{R}_B(\mathbf{1}_1, -)} \\ \mathbf{RAct}_{A_1} & \xrightarrow{F_*} & \mathbf{RAct}_{B_1} \end{array}$$

Since F preserves all the structure of $\mathbf{R}_A$, as do the embeddings of $\mathbf{R}_A$ and $\mathbf{R}_B$ into $\mathbf{RAct}_{A_1}$ and $\mathbf{RAct}_{B_1}$, one would expect F_* to preserves the structure of the full subcategory $\mathbf{R}_A$ of $\mathbf{RAct}_{A_1}$, including U_A, its finite products and their exponentials. This is another way to argue that $\bar{F}$ is indeed a morphism of λ-theories.

6.5.4 The unit

Definining the unit of the adjunction boils down to a version of Scott's representation theorem.

Lemma 6.41. *For a λ-theory L, we can define a λ-theory isomorphism*

$$\eta_L : L \xrightarrow{\sim} E(U_{L_0}).$$

Proof. Recall from Remark 6.28 that we have a chain of equivalences

$$\mathbf{Pshf}_L \xrightarrow{\sim} P\mathbf{L} \xrightarrow{\sim} PC_{L_1} \xrightarrow{\sim} PC_{L_{01}} \xrightarrow{\sim} \mathbf{RAct}_{L_{01}}.$$

It sends the presheaf L^n to the set L_1^n with a right action sending $s : L_1^n$ and $t : (L_0)_1$ to $(s_i \bullet_L \rho(t))_i : L_1^n$. We have an isomorphism $s \mapsto (\lambda(s_i))_i : (L_1)^n \to U_{L_0}$, with inverse $(\rho_i)_i : s \mapsto (\rho(s_i))_i$.

Then η_L arises by combining this with the isomorphism from Scott's representation theorem (Theorem 6.1)

$$\eta_L : L \xrightarrow{\sim} E_{\mathbf{Pshf}_L}(L) \xrightarrow{\sim} E_{\mathbf{RAct}_{L_{01}}}(U_{L_0}).$$

It is quite easy to work out that we can make this explicit as (dropping the n and just writing η_L)

$$\eta_n : L_n \xrightarrow{f \mapsto ((s:\mathbf{Pshf}_{L_m^n}) \mapsto f \bullet s)_m} \mathbf{Pshf}_L(L^n, L) \xrightarrow{f \mapsto f_1} \mathbf{RAct}_{(L_0)_1}(L_1^n, L_1), \xrightarrow{\lambda \circ - \circ (\rho_i)_i} \mathbf{RAct}_{(L_0)_1}(U_{L_0}^n, U_{L_0}),$$

so $\eta_n(f)(s) = \lambda(f \bullet (\rho(s_i))_i)$ for $f : L_n$ and $s : U_{L_0}^n$. □

Remark 6.42. Note that we can do the same with Scott's version of his representation theorem, using the chain of equivalences and an embedding

$$\mathbf{R} \xrightarrow{\sim} \mathbf{R}_{L_0} \xrightarrow{\not\perp} P\mathbf{R}_{L_0} \xrightarrow{\sim} PC_{L_{01}} \xrightarrow{\sim} \mathbf{RAct}_{L_{01}},$$

Here, it is a bit harder to get an explicit formula for η, because we need to do a bit more conversion between U_A^n and the image of U^n after the embedding $\varphi : \mathbb{R} \hookrightarrow \mathbf{RAct}_{L_{01}}$. If we quickly define λ-terms

$$\langle x_i \rangle_i = \langle x_1, \dots, x_n \rangle = \langle \langle \dots \langle (\lambda x_{n+1} x_{n+2}, x_{n+2}), x_1 \rangle, \dots \rangle, x_n \rangle$$

and corresponding tuples $(x_i)_i$ and projections π_i, we have

$$L_n \xrightarrow{f \mapsto \lambda(f \bullet (\pi_i x_1)_i)} \mathbb{R}(U^n, U) \xrightarrow{f \mapsto f \circ -} \mathbf{RAct}_{L_{01}}(\varphi(U^n), U_{L_0}) \xrightarrow{(s \mapsto \langle s_i \rangle_i) \circ -} \mathbf{RAct}_{L_{01}}(U_{L_0}^n, U_{L_0}).$$

and in the end, we obtain the very same explicit formula

$$\eta_n(f)(s) = \iota_{0,n}(f)(x_i)_i \circ (\langle s_i \rangle_i) = \lambda(f \bullet (\rho(s_i))_i).$$

Lemma 6.43. *η is natural in L. That is, for all $F : \mathbf{LamTh}(L, L')$, the following diagram commutes:*

$$\begin{array}{ccc} L & \xrightarrow{\quad F \quad} & L' \\ \downarrow{\scriptstyle \eta_L} & & \downarrow{\scriptstyle \eta_{L'}} \\ E_{\mathbf{RAct}_{L_{01}}}(U_{L_0}) & \xrightarrow{\bar{F}_0} & E_{\mathbf{RAct}_{L'_{01}}}(U_{L'_0}) \end{array}$$

Proof. We must show

$$\eta_L \cdot \bar{F}_0 = F \cdot \eta_{L'}.$$

Note that for all $s : L_0$,

$$\begin{aligned} \bar{F}_{00}(\eta_L(s)) &= \gamma_{F_0}^{-1} \cdot F_{0*}(\star \mapsto (\lambda x_1, \iota_{0,1}(s))) \cdot \beta_F \\ &= \star \mapsto F_0(\lambda x_1, \iota_{0,1}(s)) \circ ((\lambda x_1 x_2, x_2) \bullet ()) \\ &= \star \mapsto (\lambda x_1, \iota_{0,1}(F_0(s))) \\ &= \eta_{L'}(F_0(s)) \end{aligned}$$

and by Lemma 6.16, this concludes the proof. □

6.5.5 The counit

Definition 6.44. For $A : \mathbf{Alg}_\Lambda$, we define a bijection $\epsilon_A : E_{\mathbf{RAct}_{A_1}}(U_A)_0 \cong A$ as

$$\epsilon_A(s) = x_1(\lambda x_2, x_2) \bullet (s(\star)) \quad \text{and} \quad \epsilon_A^{-1}(a)(\star) = (\lambda x_2, x_1) \bullet a.$$

These are inverses because $s(\star)$ is A_1-equivariant (Lemma 2.55), and then

$$(\lambda x_2, x_1(\lambda x_3, x_3)) \bullet s(\star) = s(\star) \circ ((\lambda x_1 x_2, x_2) \bullet ()) = s(\star).$$

We want to show that ϵ_A is an isomorphism of Λ-algebras. We use Lemma 4.44 for this, so we need to show that it preserves the application and the Λ-definable constants.

Lemma 6.45. *We have for all $a, b : E(U_A)_0$,*

$$\epsilon_A((x_1 x_2) \bullet (a, b)) = (x_1 x_2) \bullet (\epsilon_A(a), \epsilon_A(b)).$$

Proof. For $a, b : E(U_A)_0$, we have, using at some point the isomorphism $\delta : \mathbf{RAct}_{A_1}(U_A^{U_A}, A_2)$,

$$\begin{aligned}\epsilon_A((x_1x_2) \bullet (a, b)) &= (x_1(\lambda x_3, x_3)(x_2(\lambda x_3, x_3))) \bullet (a(\star), b(\star)) \\ &= (x_1x_2) \bullet (\epsilon_A(a), \epsilon_A(b))\end{aligned}$$

and this concludes the proof. □

To show that ϵ preserves the Λ-definable constants, we first need to show two properties of ϵ and η:

Lemma 6.46. *ϵ is natural in A. That is, for all $F : \mathbf{Alg}_\Lambda(A, B)$, the following diagram commutes:*

$$\begin{array}{ccc} E(U_A)_0 & \xrightarrow{\bar{F}_0} & E(U_B)_0 \\ \downarrow{\scriptstyle \epsilon_A} & & \downarrow{\scriptstyle \epsilon_B} \\ A & \xrightarrow{F} & B \end{array}$$

Proof. The functor $F_* : \mathbf{RAct}_{A_1} \to \mathbf{RAct}_{B_1}$ sends $X : \mathbf{RAct}_{A_1}$ to $X \times B_1/\sim : \mathbf{RAct}_{B_1}$. We have isomorphisms

$$\beta : F_*(U_A) \xrightarrow{\sim} U_B \quad \text{and} \quad \gamma : F_*(I) \xrightarrow{\sim} I,$$

with

$$\beta(a, b) = F(a) \circ b \quad \text{and} \quad \gamma^{-1}(\star) = (\star, (\lambda x_1 x_2, x_2) \bullet ()).$$

Then $\bar{F} : \mathbf{RAct}_{A_1}(I, U_A) \to \mathbf{RAct}_{B_1}(I, U_B)$ is given by

$$\bar{F}(s)(\star) = \gamma^{-1} \cdot (s \times \mathrm{id}_{B_1}) \cdot \beta = (\lambda x_2, x_1(\lambda x_3, x_3)) \bullet F(s(\star)),$$

so

$$(\bar{F} \cdot \epsilon_B)(s) = (x_1(\lambda x_2, x_2)) \bullet F(s(\star)) = (\epsilon_A \cdot F)(s),$$

which concludes the proof. □

Lemma 6.47. *ϵ and η satisfy one of the zigzag identities.*

Proof. In this case, the zigzag identity on $L \mapsto L_0$ boils down to the following diagram commuting for all L : **LamTh**:

$$\begin{array}{ccc} L_0 & \xrightarrow{\mathrm{id}_{L_0}} & L \\ & {\scriptstyle \eta_{L_0}} \searrow \quad \nearrow {\scriptstyle \epsilon_{L_0}} & \\ & E(U_{L_0})_0 & \end{array}$$

Now, note that for all $f : L_0$,

$$(\eta_{L0} \cdot \epsilon_{L_0})(f) = (x_1(\lambda x_2, x_2)) \bullet (\lambda x_1, \iota_{0,1}(f)) = f,$$

which shows that the diagram commutes. □

Now, finally, we are ready to show that ϵ is an isomorphism of Λ-algebras:

Lemma 6.48. *We have for all $s : \Lambda_0$,*

$$\epsilon_A(s \bullet ()) = s \bullet ().$$

Proof. Consider the following diagram, with $F : \mathbf{Alg}_\Lambda(\Lambda_0, A)$ given by $F(s) = s \bullet ()$:

$$\begin{array}{ccc} E(U_{\Lambda_0})_0 & \xrightarrow{\bar{F}_0} & E(U_A)_0 \\ {\scriptstyle \eta_0}\uparrow \downarrow{\scriptstyle \epsilon_{\Lambda_0}} & & \downarrow{\scriptstyle \epsilon_A} \\ \Lambda_0 & \xrightarrow{F} & A \end{array}$$

By Lemma 6.46, the square commutes and by Lemma 6.47, we have $\eta_0 \cdot \epsilon_{\Lambda_0} = \mathrm{id}_{\Lambda_0}$.

Recall that there exists a unique morphism $\iota_\Lambda : \mathbf{LamTh}(\Lambda, E(U_A))$, and that for all $s : \Lambda_0$, by definition $s \bullet_{E(U_A)_0} () = \iota_\Lambda(s) \bullet_{E(U_A)} () = \iota_\Lambda(s)$. Since we have $\eta \cdot \bar{F} : \mathbf{LamTh}(\Lambda, E(U_A))$, we must have $\iota_\Lambda = \eta \cdot \bar{F}$ and

$$\begin{aligned} \epsilon_A(s \bullet_{E(U_A)} ()) &= \epsilon_A(\bar{F}_0(\eta_0(s))) \\ &= F(\epsilon_{\Lambda_0}(\eta_0(s))) \\ &= F(s) \\ &= s \bullet_A (), \end{aligned}$$

which concludes the proof. □

6.5.6 The equivalence

By Lemma 3.2 in [nLa24a] and Lemma 6.47, η and ϵ satisfy both zigzag identities, and we can state the fundamental theorem of the λ-calculus:

Theorem 6.49. *There is an adjoint equivalence* $\mathbf{LamTh} \cong \mathbf{Alg}_\Lambda$*, sending a λ-theory L to the Λ-algebra L_0, with an inverse functor that sends a Λ-algebra A to the theory $E_{\mathbf{RAct}_{A_1}}(U_A)$.*

Remark 6.50. Hyland remarks that the isomorphism $U_A^{U_A} \cong A_2$ can be generalized to isomorphisms $U_A^{U_A^n} \cong A_{n+1}$.

$$\begin{aligned} \mathbf{RAct}_{A_1}(U_A^n, U_A) &\cong \mathbf{RAct}_{A_1}(I, U_A^{U_A^n}) \\ &\cong \mathbf{RAct}_{A_1}(I, A_{n+1}) \\ &\cong \{f : A_{n+1} \mid \forall a : A_1, f \circ a = f\} \\ &\cong A_n. \end{aligned}$$

Explicitly, we get an isomorphism $\psi : \mathbf{RAct}_{A_1}(U_A^n, U_A) \xrightarrow{\sim} A_n$, given by

$$\psi(f) = (\lambda x_2 \dots x_{n+1}, x_1(x_2, \dots, x_{n+1})) \bullet (f(\pi_1, \dots, \pi_n))$$

and

$$\psi^{-1}(g)(a_1, \dots, a_n) = (\lambda x_{n+2}, x_1(x_2 x_{n+2}) \dots (x_{n+1} x_{n+2})) \bullet (g, a_1, \dots, a_n).$$

Note that to show this, we need to use the A_1-equivariance of f at some point:

$$\begin{aligned} &f(\pi_2, \dots, \pi_{n+1})((\lambda x_{n+1}, x_{n+1}), x_1, \dots, x_n) \\ &= (f(\pi_1, \dots, \pi_n) \circ \langle \pi_2, \dots, \pi_{n+1} \rangle)((\lambda x_{n+2}, x_{n+2}), x_1, \dots, x_n) \\ &= f(\pi_1, \dots, \pi_n)(x_1, \dots, x_n). \end{aligned}$$

This gives a λ-theory structure on $(A_n)_n$, with

$$\begin{aligned} y_{n,i} &= (\lambda x_1 \dots x_n, x_{n,i}) \bullet (); \\ f \bullet g &= (\lambda x_{m+2} \dots x_{m+n+1}, x_1(x_2 x_{m+2} \dots x_{m+n+1}) \dots (x_{m+1} x_{m+2} \dots x_{m+n+1})) \bullet (f, g_1, \dots, g_m); \\ \rho(f) &= \mathbf{1}_m \circ f; \\ \lambda(h) &= f. \end{aligned}$$

for $f : A_m$, $g : A_n^m$ and $h : A_{m+1}$.

Then any $F : \mathbf{Alg}_\Lambda(A, B)$ gives functions $F : A_n \to B_n$, and these give a λ-theory morphism in $\mathbf{LamTh}((A_n)_n, (B_n)_n)$.

The natural isomorphism $\epsilon_A : A_0 \to A$ is just id_A. Note that, even though $A_0 = A$ as sets, their Λ-algebra structures are defined differently, so it takes some work to show that ϵ_A is a Λ-algebra morphism. Also, $\eta_{Ln} : L_n \to (L_0)_n$ is given by λ^n, with ρ^n as its inverse. The zigzag identities are trivial, so this gives another, very elemental proof of the fundamental theorem.

6.6 The Theory of Extensions

The fundamental theorem of the λ-calculus that Hyland shows is actually not of the form shown above. To get there, we first need to show that the category of T-algebras for an algebraic theory T has coproducts, and define the 'theory of extensions'.

Let $[\![n]\!]$ denote the finite set $\{1, 2, \ldots, n\}$. For T an algebraic theory, let $\mathbf{L}$ be its corresponding Lawvere theory (Lemma A.6).

Lemma 6.51. *Let T be an algebraic theory. The category of T-algebras has coproducts.*

Proof. This is shown in [ARV10], in the lemmas leading up to Theorem 4.5.

Explicitly, we can express the coproduct of algebras, and especially its set, as the following coend ([Hyl17], Proposition 2.5) (see also Section 2.10 for more on coends)

$$A + B = \int^{(m,n):\mathbf{L}\times\mathbf{L}} T_{m'+n'} \times A^m \times B^n,$$

considering A as a covariant functor on $\mathbf{L}$ (see Lemma A.7) and the theory presheaf T as a presheaf (see Lemma A.8).

Note that we do not need the exact definition of $A+B$ for the rest of this section. Nonetheless, it is interesting to see how it is defined and why this definition works.

One can think of $A + B$ consisting of elements $t \bullet (a + b)$ for $t : T_{m+n}$, $a : A^m$ and $b : B^n$ (writing $(a+b)$ for $(a_1, \ldots, a_m, b_1, \ldots, b_n)$), 'substituting' the a_i and b_j for the x_i and x_{m+j} in t.

However, the coend is a quotient of $\coprod T_{m+n} \times A^m \times B^n$ along some relations. These relations then give 'associativity' of this substitution $t \bullet (a + b)$. In particular, they assure that reordering or duplicating x_i and their corresponding a_i and b_j do not yield different elements. For $f : \mathbf{L}(m, m') = T_m^{m'}$, $g : \mathbf{L}(n, n') = T_n^{n'}$, associating the images on the left and right of

$$T_{m'+n'} \times A^{m'} \times B^{n'} \leftarrow T_{m'+n'} \times A^m \times B^n \to T_{m+n} \times A^m \times B^n$$

gives

$$t \bullet ((f_i \bullet a)_i + (g_j \bullet b)_j) = (t \bullet ((f_i \bullet (x_{m+n,j})_j)_i + (g_i \bullet (x_{m+n,m+j})_j)_i)) \bullet (a + b)$$

for $t : T_{m'+n'}$, $a : A^m$ and $b : B^n$.

Then it becomes clear what the action of T will be on $A+B$, although the precise definition looks complicated because we have to juggle a bit with the variables in the different T_{m+n}. For $s : T_l$, $t_i : T_{m_i+n_i}$, $a_i : A^{m_i}$ and $b_i : B^{n_i}$, define the disjoint embeddings

$$d_i : [\![m_i + n_i]\!] \cong [\![m_i]\!] \sqcup [\![n_i]\!] \hookrightarrow \left[\!\!\left[\sum_j m_j\right]\!\!\right] \sqcup \left[\!\!\left[\sum_j n_j\right]\!\!\right] \cong \left[\!\!\left[\sum_j m_j + \sum_j n_j\right]\!\!\right],$$

which we will use to make sure that the x_j in the different t_i are mapped to distinct variables. Then we can define

$$s \bullet (t_i \bullet a_i + b_i) = (s \bullet (t_i \bullet (x_{d_i(j)})_j)_i) \bullet (a_1 + b_1 + \cdots + a_l + b_l).$$

More formally (using the coend injections $A^{\sum_k m_k} \times B^{\sum_k n_k} \times T_{\sum_k m_k + \sum_k n_k} \to A + B$), this gives functions

$$T_l \to (A^{m_1} \times B^{n_1} \times T_{m_1+n_1} \to (\cdots \to (A^{m_l} \times B^{n_l} \times T_{m_l+n_l} \to A + B) \dots)),$$

commuting with the relations between the different $(A^m \times B^n \times T_{m+n})$, which, by repeatedly using the universal property of the coend, then correspond to functions

$$T_l \to (A + B \to (\cdots \to (A + B \to A + B) \dots)),$$

or, equivalently, a function

$$T_l \times (A + B)^l \to A + B.$$

We have left and right injections $A \to A + B$ and $B \to A + B$, given respectively by the maps $A^1 \times B^0 \times T_{1+0} \to A + B$ and $A^0 \times B^1 \times T_{0+1} \to A + B$:

$$a \mapsto x_1 \bullet a \quad \text{and} \quad b \mapsto x_1 \bullet b$$

and every element $A + B$ arises by the action of t on combinations of these embedded elements:

$$t \bullet (a + b) = t \bullet ((x_1 \bullet a_i)_i + (x_1 \bullet b_j)_j),$$

which ultimately can be used to show that $A + B$ indeed has the universal property of the coproduct. □

Definition 6.52. Let T be an algebraic theory and A a T-algebra. We can define an algebraic theory T_A called the *theory of extensions* of A' with $(T_A)_n = A + T_n$. The right injection of the variables $x_i : T_n$ gives the variables.

For $h : (A + T_n)^m$, sending $g : T_m$ to $g \bullet h$ gives a T-algebra morphism $T_m \to T_n + A$. Together with the right injection morphism of A into $T_n + A$, this gives us a T-algebra morphism from the coproduct: $T_m + A \to T_n + A$. Doing this for every $h : (A + T_n)^m$ gives us the substitution $(T_m + A) \times (T_n + A)^m \to T_n + A$.

Showing that this is indeed an algebraic theory involves invoking the universal property of the coproduct and using properties of T-algebras and T-algebra morphisms.

Remark 6.53 (`extensions_theory`). We can turn the map $A \mapsto T_A$ into a functor $T_- : \mathbf{Alg}_T \to \mathbf{AlgTh}$. For a morphism $f : \mathbf{Alg}_T(A, B)$, we get maps

$$f + \mathrm{id}_{T_n} : \mathbf{Alg}_T(A + T_n, B + T_n).$$

We can combine these into a morphism $T_f = (f + \mathrm{id}_{T_n})_n$, and this makes T_- into a functor from $\mathbf{Alg}_T$ to $\mathbf{AlgTh}$.

To actually show that T_f is a morphism and that T_- is a functor, we use the properties of the coproduct a couple of times, as well as the fact that $f + \mathrm{id}_{T_n} : \mathbf{Alg}_T(A + T_n, B + T_n)$ preserves the T-action.

Remark 6.54. Note that for a T-algebra morphism $f : A \to B$, we have morphisms $f : A + T_n \xrightarrow{f+\mathrm{id}_{T_n}} B + T_n$.

Remark 6.55. Note that the right embeddings $r_n : T_n \to A + T_n$ give an algebraic theory morphism $(r_n)_n : T \to T_A$, so we can think of T as lying inside T_A.

The following result explains why we are interested in the theory of extensions:

Lemma 6.56 (`algebra_coslice_equivalence`)**.** *For T an algebraic theory and A a T-algebra, we have an adjoint equivalence $\mathbf{Alg}_{T_A} \cong (A \downarrow \mathbf{Alg}_T)$ between algebras for T_A and the coslice category under A.*

Proof. Let B be a T_A-algebra. Pullback along the embedding $(r_n)_n : T \to T_A$ gives $(r_n)_n^*(B) : \mathbf{Alg}_T$. Also, we have a T-algebra morphism $A \to B$ given by the composition

$$A \to (T_A)_0 \xrightarrow{f \mapsto f \bullet ()} B.$$

Conversely, take $f : \mathbf{Alg}_T(A, B)$. For $b : B^n$, we have a T-algebra morphism $T_n \to B$ given by $f \mapsto f \bullet b$. This, together with f, gives a morphism from the coproduct $A + T_n \to B$, and doing this for every $b : B^n$ gives a T_A-action on B as functions $(A + T_n) \times B^n \to B$.

Now, showing that the function $A \to (T_A)_0 \to B$ defined above is indeed a T-algebra morphism and that the other B, together with the given T_A-action is indeed a T_A-algebra, and furthermore showing that these extend to functors that together form an adjoint equivalence, involves checking a lot of details. One can indeed check that all of this holds, using the properties of algebraic theories, algebras, algebra morphisms and coproducts, as well as the fact that for all $b : B^n$, $f \mapsto f \bullet b$ is a morphism in $\mathbf{Alg}_T(A + T_n, (r_n)_n^*(B))$. However, for the sake of brevity, we will omit these and point to the formalization for the details. □

Example 6.57. Take $T_n = Z[X_1, \ldots, X_n]$, the polynomial ring (the free commutative ring) in n variables. T-algebras are equivalent to commutative rings, so we can call T 'the theory of commutative rings'. Now, for a commutative ring R, $\mathbf{Alg}_{T_R}$ is equivalent to the coslice category $(R \downarrow \mathbf{Alg}_T)$, which is the category of R-algebras. Therefore, the theory of extensions T_R can be considered to be the theory of R-algebras.

Lemma 6.58 (`factorization`)**.** *Any algebraic theory morphism f : $\mathbf{AlgTh}(S, T)$ factors through the embedding of S into the theory of extensions of the pullback of the algebra T_0:*

$$S \longrightarrow S_{f^*(T_0)} \longrightarrow T \qquad (S \to T \text{ via } f)$$

Proof. For any n, we have a map of S-algebras $[t \mapsto t \bullet (), f_n] : f^*(T_0) + S_n \to f^*(T_n)$, given on $f^*(T_0)$ by $t \mapsto t \bullet ()$ and on S_n by f_n. By the universal property of the coproduct, the following diagram commutes for all n:

$$S_n \xrightarrow{r_n} f^*(T_0) + S_n \xrightarrow{[t \mapsto t \bullet (), f_n]} T \qquad (S_n \to T \text{ via } f_n)$$

which shows that $(r_n)_n \cdot [f \mapsto f \bullet (), f_n]_n = f$.

Now, to show that $[f \mapsto f \bullet (), f_n]_n$ indeed constitutes an algebraic theory morphism is a bit more work. It involves using the universal property of the coproduct a couple of times, as well as showing that for all $g : S^n_{f^*(T_0)}$,

$$s \mapsto ([t \mapsto t \bullet (), f_m]x) \bullet_T ([t \mapsto t \bullet (), f_n]g_i)_i$$

is a morphism in $\mathbf{Alg}_S(f^*(T_0) + T_m, f^*(T_n))$. For more details, we again point to the formalization. □

Now, given a Λ-algebra A, applying the above to the initial morphism $\iota_\Lambda : \Lambda \to E_{\mathbf{R}_A}(U)$, we get the following diagram:

$$\begin{array}{ccccc} \Lambda & \longrightarrow & \Lambda_{E_{\mathbf{R}_A}(U)_0} & \xrightarrow{[t \mapsto t \bullet_{E_{\mathbf{R}_A}(U)} (), \iota_{\Lambda_n}]_n} & E_{\mathbf{R}_A}(U) \\ & & \downarrow{\scriptstyle \Lambda_{\epsilon_A}} & & \\ & & \Lambda_A & & \end{array}$$

For the final form of the fundamental theorem, we need to show that

$$\Lambda_{\epsilon_A^{-1}} \cdot [t \mapsto t \bullet_{E_{\mathbf{R}_A}(U)} (), \iota_{\Lambda_n}]_n = [a \mapsto \epsilon_A^{-1}(a) \bullet_{E_{\mathbf{R}_A}(U)} (), \iota_{\Lambda_n}]_n$$

is an isomorphism of algebraic theories. By Lemma 6.58, this is equivalent to its pullback

$$[a \mapsto \epsilon_A^{-1}(a) \bullet_{E_{\mathbf{R}_A}(U)} (), \iota_{\Lambda_n}]_n^* : \mathbf{Alg}_{E_{\mathbf{R}_A}(U)} \to \mathbf{Alg}_{\Lambda_A}$$

being an equivalence of categories.

Lemma 6.59. *The isomorphisms $\epsilon_A : E_{\mathbf{R}_A}(U)_0 \xrightarrow{\sim} A$ form a natural transformation. That is, for all $h : \mathbf{Alg}_\Lambda(A, B)$, the following diagram commutes:*

$$\begin{array}{ccc} E_{\mathbf{R}_A}(U)_0 & \xrightarrow{h} & E_{\mathbf{R}_B}(U)_0 \\ \downarrow{\scriptstyle \epsilon_A} & & \downarrow{\scriptstyle \epsilon_B} \\ A & \xrightarrow{h} & B \end{array}$$

Proof. This follows from simple unfolding and using the property of algebra morphisms: For all $f : E_{\mathbf{R}_A}(U)_0 = \{f : A \mid (x_1 \circ I_c) \bullet_A f = f\}$,

$$\begin{aligned} \epsilon_B(h(f)) &= (x_1 c_1) \bullet_B h(f) \\ &= h((x_1 c_1) \bullet_A f) \\ &= h(\epsilon_A(f)). \end{aligned}$$

□

Lemma 6.60. *The pullback functor*

$$[a \mapsto \epsilon_A^{-1}(a) \bullet_{E_{\mathbf{R}_A}(U)} (), \iota_{\Lambda_n}]_n^*$$

is essentially surjective.

Proof. Take $B : \mathbf{Alg}_{\Lambda_A}$. By Lemma 6.56 we can consider B to be a object in the coslice category $h : \mathbf{Alg}_\Lambda(A, B)$. As shown in Section 6.4, h sends elements in $\mathbf{R}_A(U^n, U)$ to elements in $\mathbf{R}_B(U^n, U)$, so we can regard it as a morphism $(h)_n : \mathbf{AlgTh}(E_{\mathbf{R}_A}(U), E_{\mathbf{R}_B}(U))$. We then have a $E_{\mathbf{R}_A}(U)$-algebra $(h)_n^*(E_{\mathbf{R}_B}(U)_0)$. Now we need to prove that we have an isomorphism of Λ_A-algebras

$$([a \mapsto \epsilon_A^{-1}(a) \bullet (), \iota_{\Lambda_n}]_n \cdot (h)_n)^*(E_{\mathbf{R}_B}(U)_0) \cong B.$$

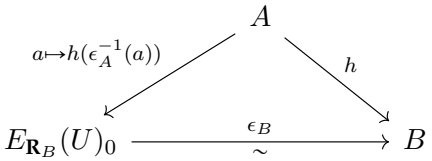
Under the equivalence in Lemma 6.56, this pullback of $E_{\mathbf{R}_B}(U)_0$ corresponds to some $(B', h') : (A \downarrow \mathbf{Alg}_\Lambda)$. The set of B' is $\mathbf{R}_B(I, U)$, its Λ-action is given by

$$(f, b) \mapsto h(\iota_{\Lambda_n}(f)) \bullet_{E_{\mathbf{R}_B}(U)} b,$$

and the morphism h' is given by $h'(a) = h(\epsilon_A^{-1}(a))$. Note that by initiality of Λ, the following diagram of algebraic theories commutes

$$\begin{array}{ccc} & \Lambda & \\ {\scriptstyle \iota_\Lambda}\swarrow & & \searrow{\scriptstyle \iota_\Lambda} \\ E_{\mathbf{R}_A}(U) & \xrightarrow{(h)_n} & E_{\mathbf{R}_B}(U) \end{array}$$

Therefore, $h(\iota_{\Lambda_n}(f)) = \iota_{\Lambda_n}(f)$, so the Λ-action on B' is exactly the action on $\iota_\Lambda^*(E_{\mathbf{R}_B}(U)_0)$, which means that $B' = \iota_\Lambda^*(E_{\mathbf{R}_B}(U)_0)$. We have the following diagram in $\mathbf{Alg}_\Lambda$:



By naturality of ϵ, this diagram commutes, which shows that ϵ_B is an isomorphism in the coslice category under A. Pulling this isomorphism back along the equivalence from Lemma 6.56 gives an isomorphism of Λ_A-algebras

$$([a \mapsto \epsilon_A^{-1}(a) \bullet (), \iota_{\Lambda_n}]_n \cdot h)^*(E_{\mathbf{R}_B}(U)_0) \cong B$$

and this concludes the proof. □

Lemma 6.61. *For ι_Λ : **LamTh**$(\Lambda, E_{\mathbf{R}_A}(U))$ and $s_i : \Lambda_n$, we have*

$$\iota_\Lambda((s_i)_i) = \langle x_i \rangle_i \bullet_A (\iota_\Lambda(s_i))_i.$$

Proof. By the recursive nature of the definitions of $(s_i)_i$ and $\langle x_i \rangle_i$, it suffices to show that for $a, b : \Lambda_n$, $\iota_\Lambda((a,b)) = \langle x_1, x_2 \rangle \bullet_A (\iota_\Lambda(a), \iota_\Lambda(b))$ and that $\iota_\Lambda(c_n) = I_c \bullet_A ()$. Since ι_Λ is defined via structural induction, this is just a matter of straightforward but tedious unfolding and rewriting, at some point using the fact that $(x_1 \circ U^n) \bullet_A \iota_\Lambda(a) = \iota_\Lambda(a)$:

$$\begin{aligned}
\iota_\Lambda((a,b)) &= \iota_\Lambda(\lambda x_{n+1}, x_{n+1}\iota_{n,n+1}(a)\iota_{n,n+1}(b)) \\
&= (\lambda x_2 x_3, x_1(x_2, x_3)) \bullet_A \iota_\Lambda(x_{n+1}\iota_{n,n+1}(a)\iota_{n,n+1}(b)) \\
&= (\lambda x_4 x_5, (\lambda x_6, x_1 x_6 (x_2 x_6)(x_3 x_6))(x_4, x_5)) \bullet_A (\iota_\Lambda(x_{n+1}), \iota_\Lambda(a \bullet_\Lambda (x_{n+1,i})_i), \iota_\Lambda(b \bullet_\Lambda (x_{n+1,i})_i)) \\
&= (\lambda x_3 x_4, \pi_{n+1,n+1}(x_3, x_4)((x_1 \circ \langle \pi_{n+1,i} \rangle)_i(x_3, x_4))((x_2 \circ \langle \pi_{n+1,i} \rangle)_i(x_3, x_4))) \bullet_A (\iota_\Lambda(a), \iota_\Lambda(b)) \\
&= (\lambda x_3 x_4, x_4((x_1 \circ \langle \pi_{n,i} \rangle_i)x_3)((x_2 \circ \langle \pi_{n,i} \rangle_i)x_3)) \bullet_A (\iota_\Lambda(a), \iota_\Lambda(b)) \\
&= (\lambda x_3 x_4, x_4(x_1 x_3)(x_2 x_3)) \bullet_A (\iota_\Lambda(a), \iota_\Lambda(b)) \\
&= \langle x_1, x_2 \rangle \bullet_A (\iota_\Lambda(a), \iota_\Lambda(b))
\end{aligned}$$

and

$$\begin{aligned}
\iota_\Lambda(c_n) &= \iota_\Lambda(\lambda x_{n+1}, x_{n+1}) \\
&= (\lambda x_2 x_3, x_1(x_2, x_3)) \bullet_A \iota_\Lambda(x_{n+1}) \\
&= (\lambda x_1 x_2, \pi_{n+1,n+1}(x_1, x_2)) \bullet_A () \\
&= (\lambda x_1 x_2, x_2) \bullet_A () \\
&= I_c \bullet_A ().
\end{aligned}$$

□

Remark 6.62. Note that for any $E_{\mathbf{R}_A}(U)$-algebra B, we can view the pullback Λ_A-algebra $[a \mapsto \epsilon_A^{-1}(a) \bullet_{E_{\mathbf{R}_A}(U)} (), \iota_{\Lambda_n}]^* B$ as an object in the coslice category under A, given by

$$a \mapsto \epsilon_A^{-1}(a) \bullet_B () : \mathbf{Alg}_\Lambda(A, \iota_\Lambda^* B).$$

Functoriality of the endomorphism theory construction of U : $\mathbf{R}_A$ and U : $\mathbf{R}_{\iota_\Lambda^* B}$, gives a λ-theory morphism

$$a \mapsto \epsilon_A^{-1}(a) \bullet_B () : \mathbf{LamTh}(E_{\mathbf{R}_A}(U), E_{\mathbf{R}_{\iota_\Lambda^* B}}(U)).$$

Then, we can pull back the theory algebra along this morphism, to get, again a $E_{\mathbf{R}_A}(U)$-algebra

$$(a \mapsto \epsilon_A^{-1}(a) \bullet_B ())^*(E_{\mathbf{R}_{\iota_\Lambda^* B}}(U)_0).$$

Lemma 6.63. *For $B : \mathbf{Alg}_{E_{\mathbf{R}_A}(U)}$, we have an isomorphism of $E_{\mathbf{R}_A}(U)$-algebras given by*

$$\epsilon_{\iota_\Lambda^* B} : (a \mapsto \epsilon_A^{-1}(a) \bullet_B ())^*(E_{\mathbf{R}_{\iota_\Lambda^* B}}(U)_0) \xrightarrow{\sim} B.$$

Proof. Note that the underlying set of $(a \mapsto \epsilon_A^{-1}(a) \bullet_B ())^*(E_{\mathbf{R}_{\iota_\Lambda^* B}}(U)_0)$ is $\mathbf{R}_{\iota_\Lambda^* B}(I, U)$, and that $\epsilon_{\iota_\Lambda^* B}$ is a bijection between this set and B. Now we only need to show that it is a morphism of $E_{\mathbf{R}_A}(U)$-algebras. Take $s : E_{\mathbf{R}_A}(U)_n$ and all $b : E_{\mathbf{R}_{\iota_\Lambda^* B}}(U)_0^n$. We have

$$\begin{aligned}
\epsilon_{\iota_\Lambda^* B}(s \bullet b) &= \iota_\Lambda(x_1 c_1) \bullet_B ((\epsilon_A^{-1}(s) \bullet_B ()) \bullet_{E_{\mathbf{R}_{\iota_\Lambda^* B}}(U)_0} b) \\
&= \iota_\Lambda(x_1 c_1) \bullet_B (\iota_\Lambda(x_1 \circ \langle x_2, \ldots, x_{n+1}\rangle) \bullet_B (\epsilon_A^{-1}(s) \bullet_B (), b_1, \ldots, b_n)) \\
&= \iota_\Lambda(x_1 c_1) \bullet_B ((\iota_\Lambda(x_1 \circ \langle x_{i+1}\rangle_i) \bullet_{E_{\mathbf{R}_A}(U)} (\epsilon_A^{-1}(s), \iota_\Lambda(x_1), \ldots, \iota_\Lambda(x_n))) \bullet_B b) \\
&= (\iota_\Lambda(x_1 c_1) \bullet_{E_{\mathbf{R}_A}(U)} (\iota_\Lambda(x_1 \circ \langle x_{i+1}\rangle_i) \bullet_{E_{\mathbf{R}_A}(U)} (\epsilon_A^{-1}(s), \iota_\Lambda(x_1), \ldots, \iota_\Lambda(x_n)))) \bullet_B b \\
&= (\iota_\Lambda(x_1(x_{i+1}c_{n+1})_i) \bullet_{E_{\mathbf{R}_A}(U)} (\epsilon_A^{-1}(s), \iota_\Lambda(x_1), \ldots, \iota_\Lambda(x_n))) \bullet_B b \\
&= (\iota_\Lambda(x_1 x_2) \bullet_{E_{\mathbf{R}_A}(U)} (\epsilon_A^{-1}(s), \iota_\Lambda((x_i c_n)_i))) \bullet_B b \\
&= ((\iota_\Lambda(x_1 x_2) \bullet_{E_{\mathbf{R}_A}(U)} (\epsilon_A^{-1}(s), \iota_\Lambda((x_i)_i))) \bullet_{E_{\mathbf{R}_A}(U)} (x_i c_n)_i) \bullet_B b
\end{aligned}$$

Also,

$$\begin{aligned}
s \bullet_B (\epsilon_B(b_i))_i &= s \bullet_B (\iota_\Lambda(x_1 c_1) \bullet_B b_i)_i \\
&= s \bullet_B (\iota_\Lambda(x_i c_n) \bullet_B (b_j)_j)_i \\
&= (s \bullet_{E_{\mathbf{R}_A}(U)} (\iota_\Lambda(x_i c_n))_i) \bullet_B b.
\end{aligned}$$

Then the result follows from the fact that

$$\begin{aligned}
\iota_\Lambda(x_1 x_2) \bullet_{E_{\mathbf{R}_A}(U)} (\epsilon_A^{-1}(s), \iota_\Lambda((x_i)_i)) &= \iota_\Lambda(x_1 x_2) \bullet_{E_{\mathbf{R}_A}(U)} ((\lambda x_2, x_1) \bullet_A s, \langle x_i\rangle_i \bullet_A (\iota_\Lambda(x_i))_i) \\
&= (\lambda x_3, x_1 x_3 x_2 x_3) \bullet_A ((\lambda x_2, x_1) \bullet_A s, \langle x_i\rangle_i \bullet_A (\pi_{n,i} \bullet_A ())_i) \\
&= (\lambda x_3, x_1 x_2 x_3) \bullet_A (s, \langle \pi_{n,i}\rangle_i \bullet_A ()) \\
&= (x_1 \circ x_2) \bullet_A (s, U^n \bullet_A ()) \\
&= s.
\end{aligned}$$

□

Lemma 6.64. *$[a \mapsto \epsilon_A^{-1}(a) \bullet (), \iota_{\Lambda_n}]_n^*$ is fully faithful.*

Proof. First of all, note that the pullback functor is always faithful, since it preserves all the 'data' (i.e. the functions) of the algebra morphisms.

Now, to show that it is full, take $B, C : \mathbf{Alg}_{E_{\mathbf{R}_A}(U)}$. Also, take a morphism

$$h : \mathbf{Alg}_{\Lambda_A}([a \mapsto \epsilon_A^{-1}(a) \bullet_B (), \iota_{\Lambda_n}]_n^* B, [a \mapsto \epsilon_A^{-1}(a) \bullet_C (), \iota_{\Lambda_n}]_n^* C).$$

By Remark 6.62, and by functoriality of the endomorphism theory construction, we get a commutative diagram

$$\begin{array}{ccc}
 & E_{\mathbf{R}_A}(U) & \\
{\scriptstyle a \mapsto \epsilon_A^{-1}(a) \bullet_B ()} \swarrow & & \searrow {\scriptstyle a \mapsto \epsilon_A^{-1}(a) \bullet_C ()} \\
E_{\mathbf{R}_{\iota_\Lambda^* B}}(U) & \xrightarrow{(h)_n} & E_{\mathbf{R}_{\iota_\Lambda^* C}}(U)
\end{array}$$

Now, using Definition 4.55 for the coslice category $(E_{\mathbf{R}_A}(U) \downarrow \mathbf{AlgTh})$, and using the previous lemma, we get the following diagram of $E_{\mathbf{R}_A}(U)$-algebras:

$$\begin{array}{ccc} (a \mapsto \epsilon_A^{-1}(a) \bullet_B ())^*(E_{\mathbf{R}_{\iota_\Lambda^* B}}(U)_0) & \xrightarrow{(a \mapsto \epsilon_A^{-1}(a) \bullet ())^* h} & (a \mapsto \epsilon_A^{-1}(a) \bullet_C ())^*(E_{\mathbf{R}_{\iota_\Lambda^* C}}(U)_0) \\ \sim \downarrow {\scriptstyle \epsilon_{\iota_\Lambda^* B}} & & \sim \downarrow {\scriptstyle \epsilon_{\iota_\Lambda^* C}} \\ B & \xrightarrow{\quad \bar{h} \quad} & C \end{array}$$

This gives us the lift

$$\bar{h} = \epsilon^{-1}_{\iota_\Lambda^* B} \cdot (a \mapsto \epsilon_A^{-1}(a) \bullet ())^* h \cdot \epsilon_{\iota_\Lambda^* C} : \mathbf{Alg}_{E_{\mathbf{R}_A}(U)}(B, C)$$

Now, when we again pull back this map to

$$[a \mapsto \epsilon_A^{-1}(a) \bullet (), \iota_{\Lambda_n}]_n^* \bar{h} : \mathbf{Alg}_{\Lambda_A}([a \mapsto \epsilon_A^{-1}(a) \bullet (), \iota_{\Lambda_n}]_n^* B, [a \mapsto \epsilon_A^{-1}(a) \bullet (), \iota_{\Lambda_n}]_n^* C).$$

Note that by naturality of ϵ, we have

$$\bar{h} = \epsilon^{-1}_{\iota_\Lambda^* B} \cdot h \cdot \epsilon_{\iota_\Lambda^* C} = \epsilon^{-1}_{\iota_\Lambda^* B} \cdot \epsilon_{\iota_\Lambda^* B} \cdot h = h$$

as functions, and therefore $[a \mapsto \epsilon_A^{-1}(a) \bullet (), \iota_{\Lambda_n}]_n^* \bar{h} = h$, so the pullback is full. □

Lemma 6.65. *Λ_A is isomorphic to $E_{\mathbf{R}_A}(U)$ as an algebraic theory.*

Proof. By Lemmas 6.60 and 6.64, the pullback functor $[a \mapsto \epsilon_A^{-1}(a) \bullet (), \iota_{\Lambda_n}]_n^*$ is a weak equivalence. Since algebra categories are univalent, this means that the pullback functor is an equivalence of categories. Then, by Lemma 4.14, $[a \mapsto \epsilon_A^{-1}(a) \bullet (), \iota_{\Lambda_n}]_n : \mathbf{AlgTh}(\Lambda_A, E_{\mathbf{R}_A}(U))$ is an isomorphism. □

Now, to show that we can replace the functor $A \mapsto E_{\mathbf{R}_A}(U)$ by $A \mapsto \Lambda_A$, we can show that the functors are isomorphic:

Lemma 6.66. *The isomorphisms $[a \mapsto \epsilon_A^{-1}(a) \bullet_{E_{\mathbf{R}_A}(U)} (), \iota_{\Lambda_n}]_n : \mathbf{AlgTh}(\Lambda_A, E_{\mathbf{R}_A}(U))$ form a natural isomorphism between the functors*

$$A \mapsto E_{\mathbf{R}_A}(U), \quad h \mapsto (h)_n \quad \textit{and} \quad A \mapsto \Lambda_A, \quad h \mapsto \Lambda_h.$$

Proof. We must show that for all $h : \mathbf{Alg}_\Lambda(A, B)$ and for all n, the following diagram of Λ-algebras commutes:

$$\begin{array}{ccc} A + \Lambda_n & \xrightarrow{[a \mapsto \epsilon_A^{-1}(a) \bullet (), \iota_{\Lambda_n}]} & \iota_\Lambda^*(E_{\mathbf{R}_A}(U)_n) \\ {\scriptstyle h + \mathrm{id}_{\Lambda_n}} \downarrow & & \downarrow {\scriptstyle h} \\ B + \Lambda_n & \xrightarrow[[a \mapsto \epsilon_B^{-1}(a) \bullet (), \iota_{\Lambda_n}]]{} & \iota_\Lambda^*(E_{\mathbf{R}_B}(U)_n) \end{array}$$

Now, by the universal property of the coproduct $A + \Lambda_n$, it suffices to check that for all $a : A$ and $t : \Lambda_n$,

$$h(\epsilon_A^{-1}(a) \bullet_{E_{\mathbf{R}_A}(U)} ()) = \epsilon_B^{-1}(h(a)) \bullet_{E_{\mathbf{R}_B}(U)} () \quad \text{and} \quad h(\iota_{\Lambda_n}(a)) = \iota_{\Lambda_n}(a).$$

The former follows from the fact that $(h)_n : \mathbf{LamTh}(E_{\mathbf{R}_A}(U), E_{\mathbf{R}_B}(U))$ respects the substitution and from the naturality of ϵ^{-1}:

$$h(\epsilon_A^{-1}(a) \bullet_{E_{\mathbf{R}_A}(U)} ()) = h(\epsilon_A^{-1}(a)) \bullet_{E_{\mathbf{R}_B}(U)} () = \epsilon_B^{-1}(h(a)) \bullet_{E_{\mathbf{R}_B}(U)} (),$$

whereas the latter follows from the initiality of Λ, and the fact that $(h)_n : E_{\mathbf{R}_A}(U) \to E_{\mathbf{R}_B}(U)$ is a λ-theory morphism. □

Remark 6.67. Now, note that for any Λ-algebra A, Λ_A is an algebraic theory. However, for our equivalence of categories, we need a functor to the category of λ-theories. By the natural isomorphism above (which respects the algebraic theory structures), we see that the objects and morphisms in the images of $E_{\mathbf{R}_-}(U)$ and Λ_- have 'the same' algebraic theory structures, and we can transfer the additional λ-theory structures from $E_{\mathbf{R}_-}(U)$ to Λ_-. With some abuse of notation, this yields a functor $\Lambda_- : \mathbf{Alg}_\Lambda \to \mathbf{LamTh}$.

The final form of Hyland's representation theorem is the following:

Theorem 6.68. *The functor that sends a λ-theory L to the Λ-algebra L_0 and the functor that sends a Λ-algebra A to the theory of extensions Λ_A form an adjoint equivalence*

$$\mathbf{LamTh} \cong \mathbf{Alg}_\Lambda.$$

Proof. By 6.49, we have an adjoint equivalence given by

$$L \mapsto L_0 \quad \text{and} \quad A \mapsto E_{\mathbf{R}_A}(U).$$

By the previous lemma, the second functor is isomorphic to $\Lambda_- : \mathbf{Alg}_\Lambda \to \mathbf{LamTh}$. Therefore, we can replace one by the other.

There are two ways to see this: We may notice that we can transfer the unit, the counit and the two zigzag identities of the adjunction along the natural isomorphism and show that this all works together. As an alternative, we can also notice that the category of λ-theories is univalent, so the functor category $\mathbf{Alg}_\Lambda \to \mathbf{LamTh}$ is univalent and the natural isomorphism between the functors is an equality, and we can replace one by the other. □

Chapter 7

The Formalization

In addition to the reading, investigating and writing mathematics that lead up to the previous chapters, this thesis project also had a formalization component. Parts of Hyland's paper were carefully written out in detail in a proof assistant and added to a library of formalized mathematics with the univalent point of view, called *UniMath*.

This chapter will give an overview of what was formalized, as well as point out and evaluate a couple of design decisions that were made.

7.1 Some Numbers

The code for this project was written over the course of about 18 months, spread over 20 pull requests: 13 with content about Hyland's paper, adding 23.361 and removing 7.620 lines of code, and 7 with only some missing category theoretical preliminaries, adding 3.291 and removing 816 lines.

7.2 Overview of the Formalized Material

The material that was formalized can be subdivided into material that is introduced or described in Hyland's paper, and category theoretical preliminaries that are necessary to make the proofs in Hyland's paper work.

The formalized parts of Hyland's paper are collected in a package in the library called *algebraic theories*. This package now consists of over 14.000 lines of code, and contains

- Definitions for algebraic theories (`algebraic_theory`), λ-theories (`lambda_theory`), algebras (`algebra`) and presheaves (`presheaf`) of algebraic theories, together with their morphisms and their categories. Proofs that the categories are univalent, that the categories of algebraic theories, λ-theories and presheaves have limits and that the categories of algebras and presheaves are fibered over the category of algebraic theories.
- The terminal algebraic theory (`one_point_theory`), the free algebraic theory on a set (`free_functor`), with as a special case the initial algebraic theory (`projections_theory`), the λ-theory Λ (`lambda_calculus_lambda_theory`), the T-presheaf structure on T and the T-algebra structure on T_n.
- The 'free object' algebraic theory T (`free_object_theory`) with a functor from $\mathbf{C}$ to $\mathbf{Alg}_T$ (`free_object_algebra_functor`). Also, for the special case where $\mathbf{C}$ is the category of monoids, a proof that this functor $\mathbf{C} \to \mathbf{Alg}_T$ is an equivalence (`monoid_algebra_equivalence`).
- The construction of the endomorphism theory $E(X)$ for some object $X : \mathbf{C}$ in a category $\mathbf{C}$ with finite products (`endomorphism_lambda_theory`).
- The original version of Scott's representation theorem (`representation_theorem_iso`).

- Hyland's version of Scott's representation theorem (`presheaf_lambda_theory_iso`), including the construction of the presheaf $A(P, 1)$ (`plus_1_presheaf`) and the proof that it is the exponential object P^T (`theory_presheaf_exponentiable`).
- The construction of a λ-theory from a Λ-algebra A as the endomorphism theory $E_{\mathbf{RAct}_{A_1}}(U_A)$, as discussed in Definition 6.32 (`lambda_algebra_theory`).
- The construction of a Lawvere theory $\mathbf{L}$ from an algebraic theory T (`algebraic_theory_to_lawvere`), and the equivalence between the presheaf category PL and the presheaf category $\mathbf{Pshf}_T$ (`algebraic_presheaf_weq_lawere_presheaf`).
- The theory of extensions T_A of a T-algebra (`extensions_theory`), the equivalence between $\mathbf{Alg}_{T_A}$ and $A \downarrow \mathbf{Alg}_T$ (`algebra_coslice_equivalence`) and the factorization of every theory morphism $f : S \to T$ through $S_{f*(T_0)}$ (`factorization`).
- An axiomatic definition of the pure λ-calculus as discussed in Section 7.10 (`lambda_calculus`).

Since Hyland's paper uses a lot of general category theory, formalizing it entailed adding the preliminaries that had not yet been formalized to the library. Among these are:

- Univalence of the Sigma displayed category (`is_univalent_sigma_disp`).
- The Sigma displayed category creates limits (`creates_limits_sigma_disp_cat`).
- The definitions $\overline{\mathbf{C}}$ (`karoubi_envelope`) and $\tilde{\mathbf{C}}$ (`karoubi_envelope'`) for the Karoubi envelope, together with some properties and their equivalence (`karoubi_equivalence`).
- The contents of Section 2.11: The functorial construction of a one-object category from a monoid (`monoid_to_category`), the equivalence between it's presheaves and sets with a right monoid action (`monoid_presheaf_action_equivalence`), properties of the category of sets with a monoid action and restriction and extension of scalars (`scalar_restriction_functor`, `scalar_extension_functor`).
- The Yoneda embedding preserves exponential objects (`yoneda_preserves_exponentials`).
- The uniqueness of the Rezk completion (`rezk_completion_unique`).
- If $F : \mathbf{C}_1 \to \mathbf{C}_2$ is a fully faithful functor, and $\mathbf{D}$ is a category with colimits, then the precomposition functor $F \bullet - : [\mathbf{C}_2, \mathbf{D}] \to [\mathbf{C}_1, \mathbf{D}]$ is split essentially surjective (`pre_comp_split_essentially_surjective`).
- A proof of a generalized version of Lemma 6.3, showing that if $F_1 : \mathbf{C}_1 \to \mathbf{C}_2$ is a functor, if $F_2 : \mathbf{C}_2 \to \mathbf{C}_3$ is a fully faithful functor, if $\mathbf{D}$ is a category with colimits and if the precomposition $(F_1 \bullet F_2) \bullet - : [\mathbf{C}_3, \mathbf{D}] \to [\mathbf{C}_1, \mathbf{D}]$ is an adjoint equivalence, then $F_1 \bullet -$ and $F_2 \bullet -$ are adjoint equivalences too (`adjoint_equivalence_1_from_comp`).
- The univalent category $\mathbf{Set}^A$ of indexed sets $(X_a)_a$ over a type A (`indexed_set_cat`), which can be defined formally as

$$\mathbf{Set}_0^A = (A \to \mathbf{Set}) \quad \text{and} \quad \mathbf{Set}^A(X, Y) = \prod_a X_a \to Y_a$$

  that is univalent (`is_univalent_indexed_set_cat`) and has limits (`limits_indexed_set_cat`).

7.3 Equality, Isomorphisms and Equivalences

One of the lessons during this project was that for formalizing, it is always important to choose te right equality for the job. For example, for two categories $\mathbf{C}$ and $\mathbf{D}$, one can aim to prove either that $\mathbf{C} = \mathbf{D}$ or that we have a functor $F : \mathbf{C} \to \mathbf{D}$ that has a right and left inverse, or that we have a functor $F : \mathbf{C} \to \mathbf{D}$ that is an adjoint equivalence. Even though for univalent categories, these three notions all coincide, in practice it is really hard to show equality of categories directly: To show $\mathbf{C} = \mathbf{D}$, we would need to show that we have an

equality $H : \mathbf{C}_0 = \mathbf{C}'_0$, and then that for all $c, c' : \mathbf{C}$

$$(\mathbf{C}(\text{transport}_H(c), \text{transport}_H(c'))) = \mathbf{D}(c, c').$$

It is usually much easier to show that there exists a functor F with an inverse, because then we do not have to show these equalities of types, but can suffice with maps between the types and equalities of the objects and morphisms that are mapped. However, for the morphisms we still have transports.

The easiest option is usually to show that we have an adjoint equivalence. That way, we do not have to show equality of objects, but only isomorphism (even though in a univalent category, that is of course the same). Then we only have to show equality of morphisms, and this is usually very doable.

An example of this is the proof that for the algebraic T where T_n is the free monoid on $\{x_1, \ldots, x_n\}$, the T-algebras are equivalent to monoids. Initially, this was a proof about (weak) equivalence (so under univalence, equality) of the object types, but of course, this says nothing about morphisms, so it is incomplete. However, it was possible to remove parts of the proof, and use the rest to construct an adjoint equivalence. Since the category in question is univalent, this adjoint equivalence gives us an equivalence of the object types for free.

In general, the formalization is the cleanest and the easiest when we use equality for elements of homotopy sets (morphisms, or terms in an algebraic theory for example), isomorphisms for objects in a category and adjoint equivalences for categories themselves.

7.4 Displayed Categories

Displayed categories (introduced in [AL17]) are a mathematical idea which provides a great tool in formalizing categories. One of the motivations behind displayed categories is the fact that mathematicians often define a category in terms of another: a monoid is a set together with a binary operation, a group is a monoid in which every element has an inverse, a topological space is a set together with a chosen collection of subsets.

Traditionally, one would say in such a case that we have a 'forgetful functor' $F : C' \to C$. However, working with displayed categories has some advantages over this older approach, both conceptually and practically.

A displayed category D over a category C firstly consists of a type D_c for every $c : C$ (corresponding to the type of objects $F^{-1}(c)$) and a type $d \to_f d'$ for all $d : D_c$, $d' : D_{c'}$ and $f : C(c, c')$ (corresponding to $C'((c, d), (c', d'))$). A displayed category also consists of an identity 'morphism' $d \to_{\text{id}_c} d$ for all $d : D_c$, and compositions $\bar{f} \cdot \bar{g} : d \to_{f \cdot g} d''$ for $\bar{f} : d \to_f d'$ and $\bar{g} : d' \to_g d''$.

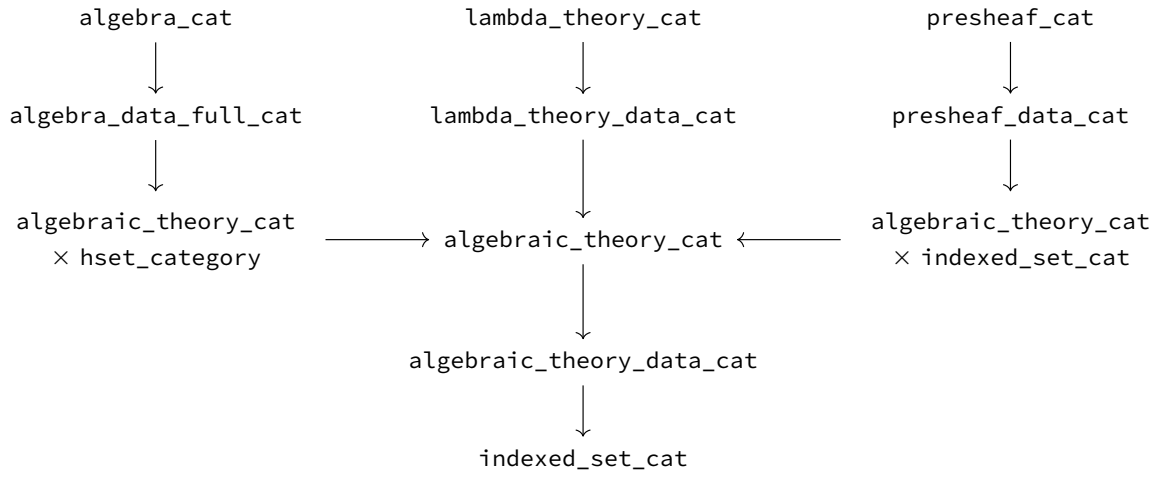
When we have a displayed category D over C, we can form the *total category* $\int_C D$ (which corresponds to the category C' in the forgetful functor example), consisting of pairs (c, d) with $d : D_c$. Then the forgetful functor is given by $\pi_1 : \int_C D \to C$. Also, for every $c : C$, we can form the *fiber category* over c, which we will also denote as Dc. The objects here are the displayed objects over c, and the morphisms are the displayed morphisms over id_c. In the forgetful functor example, this is the preimage category $F^{-1}(c)$. Lastly, note that if we have a displayed category D over C, and a displayed category E over $\int_C D$, then we can form the *sigma displayed category* $\sum_D E$ over C, where $(\sum_D E)_c$ consists of pairs (d, e) with $d : D_c$ and $e : E_{(c,d)}$. Of course, the total categories are equivalent: $\int_{\int_C D} E \cong \int_C (\sum_D E)$, which boils down to rebracketing $((c, d), e)$ to $(c, (d, e))$.

One of the reasons why displayed categories are so useful when formalizing, is the fact that for every next category, only the 'new' parts have to be defined. If we define a category that consists of 'functors with some additional data', we do not again have to show what the identity and composition functors are, but instead it suffices to construct the additional data for those functors. Also, displayed categories come in handy because some properties of $\int D$

that are hard to prove directly, can be derived from properties of C and the fiber categories D_c. For example, we can show that the category of groups is univalent, by noting that **Set** is univalent, and by furthermore showing that the fibers of the displayed category of monoids over **Set** and the fibers of the displayed category of groups over the category of monoids, are univalent.

7.4.1 The categories in question

Here is a diagram of the displayed structure of the categories of algebraic theories, λ-theories, algebraic theory algebras and algebraic theory presheaves, every arrow denotes that a category is displayed over the total category of the next one.



In the end, they all derive from the category of indexed sets over the natural numbers (or equivalently, the category of sequences of sets).

Note that every category here is constructed in two layers: in the first layer the 'data category', the displayed objects give the structure (the algebraic theory variables and substitution, the algebra action or the presheaf action) of the objects in question, and the displayed morphisms preserve this structure. In the second step, we take the full subcategory, consisting of the objects that satisfy the right properties. For algebraic theories, these properties are the axioms about the interaction between the substitution and the variables.

Note that the algebra data and presheaf data categories are displayed over a product category, which is displayed over the category of algebraic theories (see Subsection 7.4.6).

The reason why we first construct the category of all algebraic theory algebras together, is because we need we need it to show that that algebras are fibered over algebraic theories (see Subsection 7.4.3). In fact, we need the category of algebras as a displayed category over the category of algebraic theories. Since in the construction given above, it is displayed over the category of algebra data, we use the sigma construction twice, to bundle all the algebra information in a displayed category of one layer over algebraic theories. Then the category of T-algebras can be defined as the fiber over T of this displayed category. For presheaves, it is the same story.

Even though the approach of first defining the categories of all algebras or presheaves and then taking a fiber of this, is necessary to talk about fibrations, there is a drawback to this approach. Morphisms in any fiber category are the displayed morphisms over id_T. Therefore, naively composing two morphisms in our fiber gives a displayed morphism over $\mathrm{id}_T \cdot \mathrm{id}_T$ and we need to transport over the equality $\mathrm{id}_T \cdot \mathrm{id}_T = \mathrm{id}_T$ to get morphisms in our fiber category again. So even though we can prove that, as expected, $(f \cdot g)(a) = g(f(a))$ for $f : \mathbf{Alg}_T(A, B)$, $g : \mathbf{Alg}_T(B, C)$ and $a : A$, this is no longer a definitional equality.

7.4.2 Univalence

All of the categories in the diagram above are univalent. The proofs of this proceed by reducing to known or easy cases. For example, we know that a full subcategory of a univalent category is again univalent because it inherits isomorphisms and equalities. Also, we know that a product of univalent categories is univalent again, since its isomorphisms and equalities are equivalent to pairs of isomorphisms and equalities of its factors. We already knew that **Set** is univalent, and in this project, the category of sets, indexed over a type, was constructed and shown to be univalent.

The most interesting proofs are for the '...data' categories. These proofs reduce to univalence of the underlying category (so we prove univalence layer by layer) and univalence of the fiber categories, so for example the fiber of all algebraic theory data structures on one indexed set $(T_n)_n$. Showing that this fiber is univalent, means showing that for any two choices of substitution functions $f_n, g_n : T_m \times T_n^m \to T_n$ and variables $x_{i,n}, y_{i,n} : T_n$, there is an equivalence between '$(f_n)_n = (g_n)_n$ and $(x_{i,n})_{i,n} = (y_{i,n})_{i,n}$' and 'the identity on T_n commutes with f_n and g_n, and with the $x_{i,n}$ and $y_{i,n}$'. Since these are mere propositions and they imply each other, we indeed have this equivalence.

7.4.3 Fibrations

One of the places where displayed categories are conceptually better to work with than forgetful functors, is in the case of fibrations. Recall that a functor $P : C' \to C$ is a fibration if for every $Y : C'$ and $f : C(X, P(Y))$, there exists $\bar{X} : C'$ with $P(\bar{X}) = X$ (and a cartesian morphism $\bar{f} : C'(\bar{X}, Y)$ with $P(\bar{f}) = f$). The equality $P(\bar{X}) = X$ is on objects in the category, and this violates the *principle of equivalence*: 'if something is true for A, and A is isomorphic to B, then it should also hold for B'. Of course, if the category is univalent, isomorphism and equality are the same, but definitions that use equality on objects still give a bit of conceptual friction.

However, we can also define this in the language of displayed categories. The definition becomes

Definition 7.1. A displayed category D over C is a fibration if for every $\bar{Y} : DY$ and $f : C(X, Y)$, we have $\bar{X} : DX$ and a cartesian morphism $\bar{f} : \bar{X} \to_f \bar{Y}$.

This avoids using equality on objects, because we can just talk about 'the objects above X'. Therefore, in UniMath fibrations are defined in this way and the algebraic theories package uses this definition to show that the displayed categories of algebraic theory algebras and presheaves, over the category of algebraic theories, are fibrations.

7.4.4 Limits

In this subsection, we will mainly treat binary products, as they are somewhat simpler to understand than limits in general. However, it is not too hard to generalize the material presented here to limits in general, and the formalized proofs treat limits in general, instead of binary products.

Recall from Remark 4.4 that algebraic theories have all limits, and that given a diagram, the underlying set of the limit is the limit of the underlying sets. Compare this to group or ring theory: for rings R and S, the set $R \times S$ can again be given a ring structure, which is the binary product of R and S in the category of rings.

This is all very reminiscent of the way that displayed categories 'borrow' information from their base category. Now if a category C has binary products, we say that a displayed category D over C *creates binary products*, if for all $\bar{X} : D_X$, $\bar{Y} : D_Y$, we can 'lift' their product.

That is, if we can find

$$\overline{X \times Y} : D_{X \times Y}, \quad \bar{\pi_1} : \overline{X \times Y} \to_{\pi_1} \bar{X} \quad \text{and} \quad \bar{\pi_2} : \overline{X \times Y} \to_{\pi_2} \bar{Y}$$

and if $(X \times Y, \overline{X \times Y})$ with the projections $(\pi_1, \bar{\pi_1})$ and $(\pi_2, \bar{\pi_2})$ is the product of $(X, \bar{X})$ and $(Y, \bar{Y})$ in $\int_C D$. By definition, if D creates binary products, then $\int_C D$ has binary products.

In itself, this is not a very revolutionary idea. However, when formalizing, this allows us to work layer by layer, every time adding a little bit of new information, and reusing the rest from the layer below to show that every category that we construct has binary products. Also, in the rest of this section we will see that very often, it suffices to formalize a lot less than the full construction of a binary product, and that when we do constructions on displayed categories, like the sigma displayed category or taking a fiber category, we often immediately deduce that the resulting (displayed) category still has (or creates) binary products.

During this project, a small lemma was added to the library, showing that for a full subcategory $\int_C D \subseteq C$, where D_c is a mere proposition for all $c : C$, if for all $(X, \bar{X}), (Y, \bar{Y}) : \int D$, we have $\overline{D_{X \times Y}}$ (the C-product of two objects in $\int D$ is again in $\int D$), then D creates binary products. Also, a lemma was added that shows that if a displayed category D over C, and a displayed category E over D, both create binary products, then $\sigma_D E$ also creates binary products.

Note that for many categories, the displayed morphisms are mere propositions. In particular, this holds for the '...data' categories of this project. For example, an algebra morphism is a function that 'respects the operation', and respecting the operation is a mere proposition. In these categories, it is not necessary to give a full proof that the object $(X \times Y, \overline{X \times Y})$ and the morphisms $(\pi_i, \overline{\pi_i})$ form a binary product, because it suffices to show that for all $C, \bar{C}, f, g$ and all $\bar{f} : \bar{C} \to_f \bar{A}$ and $\bar{g} : \bar{C} \to_g \bar{B}$, we can lift the product morphism to

$$\overline{\langle f, g \rangle} : \bar{C} \to_{\langle f,g \rangle} \overline{A \times B}$$

For example, if the function f commutes with the algebra actions on $\bar{C}$ and $\bar{A}$, and g commutes with the actions on $\bar{C}$ and $\bar{B}$, then we need to show that the product morphism $\langle f, g \rangle$ commutes with the actions on $\bar{C}$ and $\overline{A \times B}$. A lemma showing that it suffices to show that we can lift the product morphism was added to the library, and used to show that the '...data' displayed categories create binary products.

Now, recall that the categories of T-algebras and T-presheaves are a fiber of a displayed category. Now, if we want a binary product of $X, Y : \mathbf{Alg}_T$ and we take the product of (T, X) and (T, Y) in the category of all algebras, we end up with $(T \times T, X \times Y)$, or $X \times Y : \mathbf{Alg}_{T \times T}$, even though we would like to have (T, X) for some $X : \mathbf{Alg}_T$. It turns out that here we need the fact that algebras and presheaves are fibered over algebraic theories, and adding the following lemma to the library gave the final brick for showing show that all categories discussed here have limits:

Lemma 7.2. *For a displayed category D over a category C, suppose that C has binary products, that D creates binary products, and that D is a fibration. Then the fiber categories D_X have binary products.*

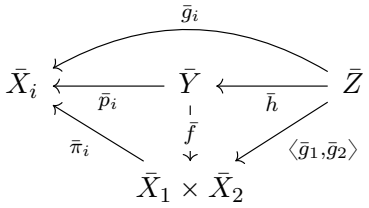
Proof. Take $\bar{X}_1, \bar{X}_2 : D_X$. We have a product $\bar{X}_1 \times \bar{X}_2 : D_{X \times X}$ with projections $\bar{\pi}_i : \bar{X}_1 \times \bar{X}_2 \to_{\pi_i} \bar{X}_i$. For the diagonal morphism on X, the fibration gives an object $\bar{Y}$ and a a cartesian lift:

$$\bar{f} : \bar{Y} \to_{\langle \mathrm{id}_X, \mathrm{id}_X \rangle} \bar{X}_1 \times \bar{X}_2,$$

so we have projections

$$\bar{p}_i = \bar{f} \cdot \bar{\pi}_i : \bar{Y} \to_{\mathrm{id}_X} \bar{X}_i.$$

In the diagram below, the first row is the fiber D_X and the second row is the fiber $D_{X \times X}$.



Now, given some $\bar{g}_1 : \bar{Z} \to_{g_1} \bar{X}_1$ and $\bar{g}_2 : \bar{Z} \to_{g_2} \bar{X}_2$, we have

$$\langle \bar{g}_1, \bar{g}_2 \rangle : \bar{Z} \to_{\langle g_1, g_2 \rangle} \bar{X}_1 \times \bar{X}_2.$$

Because $\bar{f}$ is cartesian, we have a unique

$$\bar{h} : \bar{Z} \to_{\mathrm{id}_X} \bar{Y},$$

such that

$$\bar{h} \cdot \bar{f} = \langle \bar{g}_1, \bar{g}_2 \rangle.$$

Therefore,

$$\bar{h} \cdot \bar{p}_i = \bar{h} \cdot \bar{f} \cdot \bar{\pi}_i = \langle \bar{g}_1, \bar{g}_2 \rangle \cdot \bar{\pi}_i = \bar{g}_i,$$

which shows that $\bar{Y}$ is the product of $\bar{X}_1$ and $\bar{X}_2$ in D_X. □

7.4.5 Chicken or egg?

The formalization started with definitions for the objects and morphisms in question: algebraic theories and algebras. Using these, the categories were defined directly. However, since showing univalence is much easier when working with displayed categories, the definitions of the categories were decoupled from the definitions of the objects and morphisms, and instead were constructed as displayed categories. This meant that of every definition, a part was duplicated: once for the object (and morphism) types, and once for the objects and morphisms in the displayed categories. This meant that it was in theory possible to get a mismatch between, for example, the definition of algebraic theories and their category.

Of course, it was possible to get rid of the objects altogether, and just define things in terms of the objects and the morphisms of the categories. However, in practice this causes problems because of *coercions*. For example, if X is an algebraic theory, mathematicians like to use the name X also to denote the sequence of sets X_n. Under the hood, this uses a coercion, which allows one to use the same symbol to denote both the entire object, or a part of it, depending on the context. However, when working with displayed categories, the category of algebraic theories is displayed over the category of algebraic theory data, which is displayed over the category of sequences of sets. It turns out that in coq, coercions on categories do not compose very well: If X is an algebraic theory, and we have coercions from algebraic theories to algebraic theory data, and from algebraic theory data to sequences of sets, we can still only use X to denote the algebraic theory data, and not the sequence of sets.

In the end, the solution was to first define the category, and then the object and morphism types as the objects and morphisms of the category. The coercions can then be defined on the standalone object and morphism types, which works very well.

7.4.6 A product of categories

Given two categories C and C', their product $C \times C'$ can be viewed as a displayed category over C (or C'), where the objects over any object are the objects of C', and the displayed morphisms $g : c \to_f c'$ are the morphisms $g : C'(c, c')$. There are two ways to formalize this.

The first approach uses *reindexing*, by taking the unique functor to the unit category $F : C \to \{\star\}$, and considering C' as a displayed category D over $\{\star\}$. Then we have the reindexed

(pullback) displayed category F^*D, with $(F^*D)_c = D_{F(c)}$, with $\int(F^*D) \cong C \times C'$. The advantage of this approach is that it uses fairly simple general machinery. The disadvantage is that for the general construction of (F^*D), we need to transport over the equalities

$$\mathrm{id}_{F(X)} = F(\mathrm{id}_X) \quad \text{and} \quad F(f) \cdot F(g) = F(f \cdot g)$$

in the definitions of, respectively, the displayed identity and composition. In practice, this adds friction to the formalization.

The second approach is by constructing the displayed category D directly, setting $D_c = C'$ and $c \to_f c' = C'(c, c')$. This is slightly more work, because we do it in an elementary way instead of using category theoretical machinery. In return, this approach gives cleaner definitions in practice for the identity and composition morphisms of the total category. Therefore, halfway during the project, a switch was made from the first to the second approach.

7.5 Duplication in Definitions

One of the adages in software engineering is 'DRY': "Don't Repeat Yourself": Long expressions and blocks of code that occur multiple times throughout the program should usually be abstracted into a separate function. One of the reasons for this is that it is easier to change code in this function, than to change every instance of the repeated expression or block of code. When writing mathematics, such functions are usually called 'lemmas'. However, in this project there was also another example of duplication, which occurred in statements of definitions and lemmas. For example, in the definition of the displayed category of algebraic theories, the displayed object type over a sequence of sets T_n is

$$\left(\prod_n \prod_{i:\{1,\dots,n\}} T_n\right) \times \left(\prod_{m,n} T_m \times T_n^m \to T_n\right),$$

corresponding to the variables and the substitution. Then, the constructor of an algebraic theory takes arguments

$$v : \prod_n \prod_{i:\{1,\dots,n\}} T_n \quad \text{and} \quad s : \prod_{m,n} T_m \times T_n^m \to T_n.$$

Also, given an algebraic theory, we have accessors:

$$x_{n,i} : T_n \quad \text{and} \quad \bullet_{m,n} : (\prod_{m,n} T_m \times T_n^m \to T_n)$$

Lastly, when we define a new algebraic theory, we need to provide terms of these types again.

Therefore, in this project, the type of every one of these components is given a name, ending in `_ax`, for 'axiom'. For example, there are `var_ax` and `subst_ax` for algebraic theories, `mor_var_ax` and `mor_subst_ax` for their morphisms, `action_ax` for algebras and `app_ax` and `abs_ax` for λ-theories. The definition of the displayed categories, the constructors, the accessors and the definitions of new objects can then refer back to this.

This indeed reduces the amount of duplication in the code, and makes it slightly easier to write some definitions, because one does not have to remember and type the exact formulation of every axiom. However, there is no free lunch here: The axioms are not immediately unfolded when they occur. Therefore, if one uses the constructor for an algebraic theory, the goals become `var_ax` and `subst_ax`, and these have to be unfolded to see what they mean. Also, when coq is asked to state the property of an algebra morphism, it responds with `mor_action_ax`, which is not very informative. This is not a big problem, per se, but it adds some friction when formalizing. This friction could be reduced a lot if coq would

have some sort of macros, which would immediately be unfolded upon use, and would never be printed.

Another drawback of this approach is that for algebras and presheaves, there are in fact two different axioms. This is because their categories are displayed over the categories of algebraic theories, and the category of T-algebras and T-presheaves are fibers of the full displayed category. Therefore, the morphisms of T-algebras are given by displayed morphisms over id_T, so the axiom of an algebra morphism g is

$$\texttt{mor\_action\_ax} : \prod_n \prod_{t:T_n} \prod_{a:A^n} g(t \bullet a) = \mathrm{id}_T(t) \bullet (g(a_i))_i,$$

and in practice, this makes it harder to work with them. There was an instance where one conversion from $\mathrm{id}_T(t)$ to t added multiple seconds to the compilation time. Therefore, for the algebras and presheaves, some of these axioms have to be stated twice: once to define the displayed category, and once for most of the other occurrences.

7.6 Tuples

A lot of the mathematics in the paper requires us to work with 'tuples': terms that bundle a certain number of terms of some type. A tuple type already occurs already as T_m^n in the definition of the substitution operation of algebraic theories:

$$\bullet : T_m \times T_n^m \to T_n.$$

Now, there are two common ways to formalize such tuples, both with their advantages and disadvantages.

The first option is to say that the type A^n just denotes the n-fold cartesian product

$$A \times (\cdots \times (A \times A) \dots).$$

There are multiple advantages to this approach. First of all, this approach allows us to easily construct 'literals', like $(5, -12) : \mathbb{Z}^2$ or $(\bot, \top, \top) : \mathrm{bool}^3$. It is also very easy to extend an n-tuple a with another element x, because this just gives the pair (x, a). Because of the clear relation between the extended tuple and the original one, this approach also allows for a nice induction principle: a way to prove things about general $a : A^n$ by proving it for $() : A^0$ and by showing that if it holds for any $a : A^m$, it also holds for $(x, a) : A^{m+1}$ for all x (note that this is a version of `fold` for lists).

The other approach is to view A^n as the type of functions from $\{1, \dots, n\}$ to A. An advantage of this is that it becomes trivial to extend a function f on A to a function on its tuples, sending a to $f \circ a$ (which would be called $(f(a_i))_i$ in this thesis). Also, this approach makes it very easy to define n-tuples of arbitrary size. For example, $(x_i)_i : T_n$ (in one of the axioms of an algebraic theory) is just the function that sends i to the variable x_i. These tuples of arbitrary size have very good computational behaviour, since the value of $(x_i)_i$ at j is, by definition, x_j.

Very early on in the project, a choice was made to use the second approach. Since many definitions require extending a mapping to a tuple, and since we do not very often need to extend tuples or do induction on a tuple, this proved to be the right choice.

The main place where we need to be able to extend tuples, is when working with λ-theories, for example in the axioms about the relation between substitution and λ-abstraction and -application. To accomplish this, we use the equivalence between $\{1, \dots, n+1\}$ and $\{1, \dots, n\} \sqcup \{\star\}$.

Also, when working with a Λ-algebra A (or an algebra) for the free monoid theory, we often want to define an operation, for example $\circ$, on A by sending a and b to $(x_1 \circ x_2) \bullet (a, b)$.

Since constructing the literal tuple (a, b) directly as a function quickly becomes a mess, we use the equivalence between $A \times \cdots \times A$ and $\{1, \ldots, n\} \to A$ to define literal tuples via the cartesian product, and then transform them to functions.

Using some lemmas about the behaviour of extended tuples and literals defined this way, this slightly mixed approach works fine. However, it would be worth investigating whether it is possible to have a tuple type in the library with all of the operations of both the function and the cartesian product approaches, so that it is no longer necessary to choose between them.

7.7 Products

A very similar problem pops up in the proof of the representation theorem, where the definition of the endomorphism theory $E_{\mathbf{Pshf}_L}(L)$ meets the isomorphism $\mathbf{Pshf}_L(L^n, L) \cong L_n$ of the Yoneda lemma for presheaves. First of all, recall that the function λ_n of $E(L)$ is defined via the following chain of morphisms

$$\mathbf{Pshf}_L(L^{n+1}, L) = \mathbf{Pshf}_(L^n \times L, L) \cong \mathbf{Pshf}_L(L^n, L^L) \xrightarrow{-\cdot\mathrm{abs}} \mathbf{Pshf}_L(L^n, L).$$

To make this easy, we would like to have $L^{n+1} = L^n \times L$, and this would give a definition $L^n = (\ldots(I \times L) \times \ldots) \times L$ where I is the terminal object.

On the other hand, recall that inverse morphism of $\mathbf{Pshf}_L(L^n, L) \cong L_n$ sends $s : L_n$ to the presheaf morphism that sends $(t_i)_i : L_m^n$ to $s \bullet t : L_m$. Here, we would like the sets of L^n to match the tuple types in the definition of the substitution $\bullet : L_n \times L_m^n \to L_m$. However, these two are incompatible, since elements of the repeated product are nested pairs, whereas the tuples in L_m^n are functions from $\{1, \ldots, n\}$ to L_m.

In this case, somewhat of a compromise was made. The endomorphism theory was indeed defined using a repeated binary product, and the construction of finite powers from the terminal object and binary products[1]. On the other hand, the Yoneda lemma for L-presheaves does indeed use the 'usual' notion of products L^n where the sets are tuple types. Luckily, the library contains a proof that two products are isomorphic, so where the definitions clash, we use the isomorphisms to translate between them. Even though this adds friction, in the end it is pretty straightforward.

Also, this project added a lemma to the library stating that in any category C, $\prod_{i:\varnothing} X_i$ (for $X : \varnothing \to T$ a family of objects) is given by the terminal object. However, the repeated binary product construction needed $\prod_{i:\{i|1\leqslant i\leqslant 0\}} X_i$, and $\varnothing$ is not definitionally equal to $\{i \mid 1 \leqslant i \leqslant 0\}$. Initially, this was solved by transporting along the equality $\varnothing = \{i \mid 1 \leqslant i \leqslant 0\}$. However, much further down the line, some proof involved an element of this product over $\{i \mid 1 \leqslant i \leqslant 0\}$, and because of the transport, coq was not able to see that this product was just $\{\star\}$. Therefore, the statement of the lemma about the empty product was changed to "For a type I and a family $X : I \to C$, if we have a function $f : I \to \varnothing$, the terminal object gives the product $\prod_{i:I} X_i$". Note that having the function f is equivalent to I being equivalent (and therefore equal) to the empty type. This change in the statement did provide the right amount of generality to make computation down the line much smoother.

[1] Unfortunately, defining the product of a tuple of objects was much harder than defining the finite power of one object. When generalizing the approach to tuples of objects, some terms suddenly do not have the 'correct' type, which has something to do with some equalities like $((x_1, \ldots, x_n) + (y))_{n+1} = y$ that do not hold definitionally. Even though this specific problem can be solved using transports, the rest of the proof becomes then much harder because the transports get in the way

7.8 The $n + p$-presheaf

Now, the definition of the endomorphism λ-theory $E_{\mathbf{Pshf}_L}(L)$ requires that the theory presheaf L is exponentiable. As shown before, P^L is given by $A(P, 1)$, the 'plus 1' presheaf, where the last variable is ignored in the L-action. In fact, one can show that P^{L^n} is given by $A(P, n)$ and this was the level of generality at which the statement was formalized. To obtain the action

$$\bullet : A(P, p)_m \times L_n^m \to A(P, p)_n,$$

the equivalence

$$\{1, \ldots, n + p\} \simeq \{1, \ldots, n\} \sqcup \{1, \ldots, p\}$$

from the library was used, which sends the first n elements to the first set, and the last p elements to the second, so this could indeed be used to construct elements of tuples A^{n+p}, separating them into first a tuple of n and then a tuple of p. After this, it was time to construct the presheaf morphisms for ρ and λ between L and $A(L, 1)$, but this turned out to be much harder than expected, because $A(L, 1)_n = L_{n+1}$, whereas ρ and λ are functions between L_n and L_{Sn}, where $Sn = 1 + n$. Unfortunately, $1 + n$ is not definitionally equal to $n + 1$, and even though we might transport along a proof $h : 1 + n = n + 1$, this will give problems down the road.

In this case, the cleanest solution was to only do the construction for the special case $p = 1$ and take $A(P, 1)_n = P_{Sn}$, using the equivalence from the library

$$s_m : \{1, \ldots, n + 1\} \simeq \{1, \ldots, n\} \sqcup \{\star\},$$

with

$$s_m(i) = \begin{cases} i & i < m \\ \star & i = m \\ i - 1 & i > m \end{cases},$$

taking $m = n$ in this case.

It is possible that constructing a new equivalence

$$\{1, \ldots, p + n\} \simeq \{1, \ldots, n\} \sqcup \{1, \ldots, p\},$$

makes it possible to generalize the result to general p again. However, this equivalence would be somewhat weird, because the order of n and p are reversed. Also, proving things about these standard finite sets is quite hard, and outside the scope of this project, so this was not attempted in this project, but potentially left for future work.

7.9 Quotients

This project used quotients twice, once explicitly and once implicitly. The first occurrence of quotients was the construction of the 'extension of scalars' functor $f_* : \mathbf{RAct}_N \to \mathbf{RAct}_M$ from Lemma 2.59. Here, the formalization worked directly with quotients over a relation. However, the relation given by $R : (xn, m) \sim (x, f(n) \cdot m)$ for all a, is reflexive and transitive, but not necessarily symmetric. Therefore, first the 'equivalence closure' $\bar{R}$ of R is taken: the smallest transitive and symmetric relation that contains R. This is then used for the quotient. To construct functions from this, like, the right action of the monoid M', we use the universal property of the quotient: a function from $X \times M'$ that sends 'related' elements (x_1, m_1) and (x_2, m_2) to the same element. However, the quotient is taken over $\bar{R}$ instead of R, so it is no longer possible to assume that $x_2 = x_1 n$ and $m_1 = f(n) \cdot m_2$ for some n. In the case of the right M'-action, it sufficed to add the following lemma to the library

Lemma 7.3. *For a relation S on a set A and a relation T on a set B, if for $f : A \to B$, we have that $S(a_1, a_2)$ implies $T(f(a_1), f(a_2))$ for all $a_1, a_2 : A$, then $\bar{S}(a_1, a_2)$ implies $\bar{T}(f(a_1), f(a_2))$ for all a_1, a_2.*

However, overall, working with these quotients in an elementary way was quite a hassle.

The second occurrence of quotients was in the proof of Corollary 2.81, or the lemma that it is based on. Initially, the statement of the lemma was only about **Set**, stating that $\iota_C \bullet - : [\bar{C}, \mathbf{Set}] \to [C, \mathbf{Set}]$ was an adjoint equivalence. The proof used coequalizers of functors, like in Lemma 2.73. Note that coequalizers of set-valued functions are given pointwise by coequalizers in **Set**, which are formalized using quotients. Because of this, when simplifying, coq usually reduced the goal to elementary statements about quotients, which somewhat obfuscated the relatively simple nature of the construction. Luckily, the construction could be made agnostic to the specific implementation of coequalizers, by introducing a variable

```
H : CoequalizersHSET
```

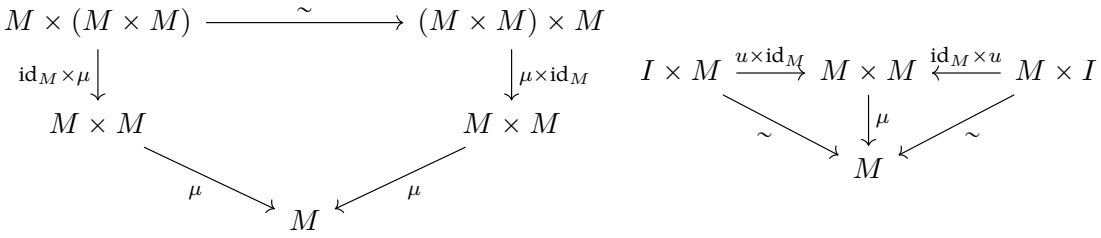
and using this instead of our known implementation. Since the proofs did not use any elements of the sets in question, it was then trivial to generalize to any category with coequalizers. It turned out that the construction was a special version of left Kan extension, which uses general colimits, and using the existing definition of Kan extension, the proof was made considerably shorter.

In hindsight, it may have been better to replace the quotient $Y := X \times M/\sim$ by a coequalizer

$$N \times X \times M \underset{(n,x,m)\mapsto(x,f(n)\cdot m)}{\overset{(n,x,m)\mapsto(xn,m)}{\rightrightarrows}} X \times M \dashrightarrow Y \ ,$$

which is closer to the way this relation is defined. Under the hood, this still takes a quotient over the equivalence closure, but now it is less visible. Here, the universal property of the coequalizer can be used to define functions out of Y, which may be easier to work with than the universal property of the quotient.

The generalization from **Set** to an arbitrary category with coequalizers, which we saw in the case of Karoubi envelopes, can also be done here, but it would take considerably more effort. The problem is that the definitions of monoids and sets with a monoid action really do use the elements of their underlying sets. For example, a monoid M has a unit u and multiplication $\cdot$, such that $u \cdot m = m = m \cdot u$ and $m \cdot (n \cdot l) = (m \cdot n) \cdot l$ for all $m, n, l : M$. The generalization for an object M in some category C, turns the unit into a function $u : I \to M$ from the terminal object and the multiplication into a function $\mu : M \times M$ from the binary product [2], such that the following diagrams commute:



Such a generalization of a monoid to a general category is called an *internal monoid*. In a similar way, sets with a monoid action can be generalized. However, this all would be a lot of

[2] Note that one does not need the full power of binary products or terminal objects to define internal monoids. A category with the product-like structure $\otimes$ and terminal-like object I that one needs to define internal monoids, is called a *monoidal category*.

work, and elementary reasoning about internal monoids of **Set** is still much more cumbersome than for the ordinary set-based monoids. Also, for the fundamental theorem we only need the ordinary set-based monoids. Therefore, this generalization was not pursued any further.

7.10 The Formalization of the λ-calculus

One of the important classes of objects in this thesis is the class of Λ-algebras: algebras for the initial λ-theory Λ, corresponding to the 'pure' λ-calculus. Therefore, much of this project required the pure λ-calculus to be formalized. Now, in this thesis, it is defined as a quotient of an inductive type. However, in the UniMath library, the use of inductive types (other than a handful of basic ones like `empty`, `bool`, `nat` and `coprod`) is prohibited [3]. Also, as we saw in section 7.9, working with quotients can be somewhat complicated.

The inductive type approach has a variation, which instead of a quotient uses a higher inductive type to force the required elements to be equal. Unfortunately, even if the use of inductive types was allowed, coq does not support higher inductive types.

Instead, a different, more axiomatic approach was taken. A class in group theory often starts with laying out a couple of axioms, like: "We have a set G, with a binary operation b, a unary operation u and an element e. b is associative, u is a left and right inverse for b and e is a unit for b." Afterwards, this structure is usually bundled into the declaration "Let G be a group." In the same spirit, every section in the formalization that needs the pure λ-calculus, starts with "Let L be the pure λ-calculus with β-equality." To this end, there is a definition `lambda_calculus`, consisting of

- A sequence of sets $(L_n)_n$;
- Variables, app, abs, subst:

  $$\mathtt{var}_{n,i} : L_n, \quad \mathtt{app}_n : L_n \times L_n \to L_n, \quad \mathtt{abs}_n : L_{Sn} \to L_n \quad \text{and} \quad \mathtt{subst}_{m,n} : L_m \times L_n^m \to L_n;$$

- A couple of identities about the interactions between the constructors. In particular, β-equality:

  $$\mathtt{subst}(\mathtt{var}_i, t) = t_i, \quad \ldots, \quad \mathtt{app}(\mathtt{abs}(s), t) = \mathtt{subst}(s, (\mathtt{var}_i)_i + (t));$$

- The induction principle, which coincides with the induction principle that a higher inductive type would have: Given, for all $t : L_n$, a type $A_{n,t}$, it is possible to construct, for every $t : L_n$, an element $f(t) : A_{n,t}$ by just giving elements and functions, corresponding to the constructors

  $$\begin{aligned} f_{\mathtt{var}(n,i)} &: A_{\mathtt{var}_{n,i}} \\ f_{\mathtt{app}(s,t)} &: A_s \times A_t \to A_{\mathtt{app}(s,t)} \\ f_{\mathtt{abs}(t)} &: A_t \to A_{\mathtt{abs}(t)} \\ f_{\mathtt{subst}(s,t)} &: A_s \times A_{t_1} \times \cdots \times A_{t_n} \to A_{\mathtt{subst}(s,t)} \end{aligned}$$

  and by showing that they are compatible with the identities on L. For example, for β-equality, this is equality between

  $$f_{\mathtt{app}}(f_{\mathtt{abs}}(s), t) : A_{\mathtt{app}(\mathtt{abs}(s),t)} \quad \text{and} \quad f_{\mathtt{subst}}(s, f_{\mathtt{var}_1}, \ldots, f_{\mathtt{var}_n}, t) : A_{\mathtt{subst}(s,(\mathtt{var}_i)_i+(t))}.$$

  Note that the left and right hand side live in different types, so the equality can only be stated using a transport over β-equality.

[3] This is because allowing inductive types requires a much larger trusted codebase of the proof assistant, and because most of mathematics can be formalized with $\sum$-types or using initial algebras of functors, instead of inductive and record types.

- A couple of identities about the interaction between induction and the constructors: if f is defined using the induction principle as above,

$$\begin{aligned} f(\mathtt{var}_i) &= f_{\mathtt{var}_i} \\ f(\mathtt{app}(s,t)) &= f_{\mathtt{app}(s,t)}(f(s), f(t)) \\ f(\mathtt{abs}(t)) &= f_{\mathtt{abs}(t)}(f(t)) \\ f(\mathtt{subst}(s,t)) &= f_{\mathtt{subst}(s,t)}(f(s), (f(t_i))_i). \end{aligned}$$

Remark 7.4. Note that the first three bullets just give L a λ-theory structure. Only by the induction principle, it becomes clear that L is the (unique) pure λ-calculus.

Remark 7.5. The induction principle has two common uses: $A_{n,t}$ can be taken to be a constant type A (or potentially A_n), in which case induction results in functions $L_n \to A$. The $A_{n,t}$ can also be taken to be mere propositions, in which case induction 'proves' something about all $t : L_n$.

In both cases, the rules about respecting the identities like β-equality become simpler: in the first case, the transports disappear, and in the second case, the rules are satisfied automatically.

Remark 7.6. Often, the pure λ-calculus is defined using just the constructors `var`, `app` and `abs`. `subst` is usually defined using induction, which gives the identities about the interaction between `subst` and the others for free. However, since this is a definition of the λ-calculus with β-equality, the induction principle must include a rule about compatibility with the β-equality. Since the definition of β-equality already uses `subst`, it is not possible to define `subst` using induction. This is why it is added as an additional constructor, along with requirements about its interaction with the other constructors. This approach is called *explicit substitution*.

7.10.1 Propagation of substitution

Once the pure λ-calculus is defined, it is not very hard to give it a λ-theory structure, using the induction principle a couple of times to show that some identities are satisfied.

Now, as mentioned in Subsection 4.4.1, any λ-theory allows the operations `var`, `app`, `abs` and `subst` with the same interaction as for the pure λ-calculus. Therefore, given any λ-calculus, it is possible to start defining more complicated structures like the $a \circ b$, $\langle a, b \rangle$ and $A \times B$ from Section 5.2, which is indeed done in the original proof of Scott's representation theorem. However the equalities about the interaction between `subst` and the other operations are not definitional, even for the pure λ-calculus. Consider the following term (using concatenation for application):

$$\lambda x_5, x_5(x_1x_2(x_4x_3)) \bullet (x_1, x_2, x_3, x_1) : L_3.$$

It is not hard to see that this results in $\lambda x_4, x_4(x_1x_2(x_1x_3))$. However, it takes a lot of steps to rewrite this: moving the substitution past the λ-abstraction, then into 4 instances of application, and lastly using 5 instances of the interaction between variables and substitution, resulting in a total of 10 rewrites for a seemingly trivial term. It is not unheard of to have 40 of these rewrites consecutively in a proof, and since there is a lot of things to prove, this quickly becomes tedious.

Therefore, a tactic was added to the project. A first version of this tactic was a variation of

```
Ltac reduce_lambda := (
  rewrite subst_var +
  rewrite subst_l_var +
```

```
  ...
  rewrite beta_equality
).
```

attempting to rewrite (once) with at least one, but possible multiple of the equalities. The statement `repeat reduce_lambda` sometimes took a couple of seconds, but this saved a lot of manual work. Still, there was much room for improvement.

Therefore, along with the original proof of Scott's representation theorem, a new version of the tactic, called `propagate_subst` has been added. It recursively traverses the λ-term in the left-hand side of the goal, checking whether the term matches a form that can be rewritten into something else. It performs the possible rewrites, and also prints these rewrite statements which can replace it. For example, if the goal is

```
(λ'm, (inflate a (inflate b var (stnweq (inr tt))))) • c =
(λ'n, (inflate (a • c) (inflate (b • c) var (stnweq (inr tt)))))
```

a call to `repeat reduce_lambda` would take about a second, but a call to `propagate_subst ()` runs in about $\frac{1}{3}$ of a second and prints

```
refine '(subst_abs _ _ _ @ _).
refine '(_ @ !maponpaths (λ x, (abs (app x _))) (inflate_subst _ _ _)).
refine '(_ @ !maponpaths (λ x, (abs (app _ (app x _)))) (inflate_subst _ _ _)).
refine '(maponpaths (λ x, (abs x)) (subst_app _ _ _ _) @ _).
refine '(maponpaths (λ x, (abs (app x _))) (subst_inflate _ _ _) @ _).
refine '(maponpaths (λ x, (abs (app _ x))) (subst_app _ _ _ _) @ _).
refine '(maponpaths (λ x, (abs (app _ (app x _)))) (subst_inflate _ _ _) @ _).
refine '(maponpaths (λ x, (abs (app _ (app _ x)))) (var_subst _ _ _) @ _).
refine '(maponpaths (λ x, (abs (app _ (app _ x)))) (extend_tuple_inr _ _ _) @ _).
```

Replacing the call to `propagate_subst` by these statements, results in the same rewrites, but these only take $\frac{1}{25}$ of a second.

Now, on top of the speedup, the new tactic is modular and extensible. It is modular, in the sense that some of its parts are also tactics themselves, and can be used for other tactics as well. For example, the `traverse` tactic, which traverses a λ-term in the goal and executes something for every subterm, is also used in a new tactic that is called `generate_refine`, which takes a pattern, and for every subterm that matches it, prints a

```
refine '(maponpaths (λ x, ... x ...) _ @ _).
```

statement, which can be used to quickly generate statements that very precisely rewrite one subterm. The `propagate_subst` tactic is also extensible, in the sense that the patterns for both the subterm traversal and the rewrites are kept in a list, which can be extended when new combinators are defined. For example, at the point where the tactics are defined, the traversal only works for the constructors `var`, `app`, `abs` and `subst`, and the rewrites only work for the interactions between these operations. Using this, composition $a \circ b$ is defined, and so a pattern to branch into a and b is added to the traversals, and a rewrite with

$$(a \circ b) \bullet t = (a \bullet t) \circ (b \bullet t)$$

is added. Progressing through the file, the same is done for combinators like the pair (a, b), the projection π_1 (including a rewrite $\pi_1\langle a, b\rangle = a$), `curry` and `n_tuple` (consisting of nested pairs).

It would be interesting to see whether parts of these tactics can be generalized. For example, whether it is possible to create an extensible tactic which shows that some type is a homotopy set, or a mere proposition, and prints the coq statements that can replace the tactic call. Or whether it would be possible to generalize `generate_refine` to very quickly generate very fast and precise rewrite statements for large and complicated goals.

7.11 The Learning Curve

Let me conclude this chapter with a more personal note about formalization. In the past fifteen years, I have worked with a plethora of different programming languages, and I would like to think that in all this time, I have learned a great deal about programming. When I start working with a new language, it takes me a couple of minutes to figure things out, and within an hour, I can probably get a small program up and running, with some help from google. However, when I started working with coq and UniMath, I kind of had to start from scratch again. As it turns out, the naive or direct approach is often not the right one when working with a proof assistant, and of the code that I wrote in the first months, almost nothing remains. Even though every programming language has some sort of learning curve, formalizing in a proof assistant is especially unforgiving: You often have to change an old definition, or start over when you are halfway through a proof, because your current approach gets bogged down further and further with, for example, unintelligible transports. For every line of code that ends up in a pull request, many lines are written and erased again.

For example, if we want to show that two types X and Y are equivalent, we need to construct an equivalence, which has the following type:

$$X \simeq Y := \{f : X \to Y \mid \forall y, \texttt{is\_contractible}(\{x \mid f(x) = y\})\}.$$

One can view this as having an 'isomorphism'

$$f : X \to Y, \quad g : Y \to X, \quad h_1(x) : g(f(x)) = x \quad \text{and} \quad h_2(y) : f(g(y)) = y,$$

but then together with a proof h_3, showing that applying f to $h_1(x)$ is the same as taking $h_2(f(x))$. This last part is to ensure that 'f is a weak equivalence' is a mere proposition, even if X and Y are not sets: if X and Y are not sets, there can be multiple paths h_1 and h_2, so f can be an isomorphism 'in multiple ways'. It can be tempting to just close our eyes and start constructing this equivalence part by part. When we then get to defining h_3, however, it is easy to get stuck, because reasoning about paths is complicated. Luckily, there is a surjection

$$\texttt{isweq\_iso} : \texttt{is\_iso}(f) \twoheadrightarrow \texttt{isweq}(f),$$

which allows us to construct an isomorphism and then get an equivalence (with h_3) for free.

Another example: When we want to define a category, usually we start by defining what the objects and morphisms in the category look like, and then constructing a category from that. Since we work in univalent foundations here, we want to show that the category is univalent, so that idtoiso : $(a = b) \to (a \cong b)$ is an equivalence. However, if we try to show this directly, the proof quickly accumulates piles of transports, that are very hard to resolve. Instead, there are two common 'indirect' approaches:

- Either we first construct multiple smaller equivalences, ignoring idtoiso for the time being:

  $$\begin{aligned}(a = b) &\simeq \{\text{componentwise equalities between } a \text{ and } b\} \\ &\simeq \{\text{componentwise isomorphisms between } a \text{ and } b\} \\ &\simeq (a \cong b),\end{aligned}$$

  and then use a proof that if we have an equivalence $f : X \simeq Y$, and we have some $g : X \to Y$ with $f(x) = g(x)$ for all $x : X$, we know that g is an equivalence as well.
- We transform the category into a stack of displayed categories, derive definitions for our objects from this displayed category, and then do the easy proofs that the fibers of every layer are univalent.

Both of these approaches are not what we would think of in the first place, but they are the approaches that turn out to work very well. In short, there is a very steep learning curve for working with a proof assistant and it takes a lot of experience to formalize mathematics. Not only to know what kind of tactics are available, but also to get a feeling for the theorems, lemmas and constructions that work well for proving or constructing something.

Chapter 8

Conclusion

In this thesis, we have seen how Dana Scott showed in an elementary way that any λ-theory arises as the endomorphism theory of a reflexive object in its category of retracts (Theorem 5.8). We saw Martin Hyland's proof that any λ-theory arises as the endomorphism theory in the presheaf category of its Lawvere theory, using the Yoneda lemma in a very elegant way (Theorem 6.1).

We also saw how Paul Taylor shows that Scott's category of retracts is relatively cartesian closed (Theorem 5.18), and that Hyland gives an interesting new proof of this (Corollary 6.9), using the pre-established fact that the presheaf category is locally cartesian closed. Here it was interesting to note that Taylor's and Hyland's proofs are about the same category in classical mathematics, but that these categories become nonequivalent in univalent foundations (Remark 6.11), one being the Rezk completion of the other (Corollary 2.75).

As we saw, there are two ways to study the λ-calculus using tools from universal algebra: both via λ-theories and Λ-algebras. We saw that Hyland gives an equivalence between these two in his Fundamental Theorem of the λ-calculus (Theorem 6.68), where part of his construction again uses a presheaf category, $\mathbf{RAct}_{A_1}$ (Theorem 6.49), parallel to his proof of Scott's representation theorem. The equivalence sends a λ-theory L to the Λ-algebra L_0 and sends a Λ-algebra A to its theory of extensions Λ_A. We also saw a couple of variations on the proof of this fundamental theorem (Theorem 6.18 and Remark 6.50), exhibiting multiple equivalent ways to construct a λ-theory from a Λ-algebra.

Lastly, we saw how part of the material in this thesis was formalized, and we evaluated the choices that were made in the formalization (Chapter 7). Unfortunately, due to the very time-consuming nature of formalization, not all of the material could be formalized. In future work, it would be interesting to see which version of Hyland's fundamental theorem would lend itself best to formalization. Personally, I would guess it is the most elementary one, exhibited in Section 6.4.

Also, since we saw that in univalent foundations, there are two nonequivalent definitions $\bar{\mathbf{C}}$ and $\hat{\mathbf{C}}$ for the category of retracts, it would be interesting to see how well Scott's and Taylor's proofs about $\bar{\mathbf{C}}$, can be made to work on its Rezk completion $\hat{\mathbf{C}}$. More generally, note that the Karoubi envelope is not specific to the material in this thesis. For example, one way to construct a cartesian closed category is by taking the Karoubi envelope of a 'semi cartesian closed category', and the cartesian closed structure on $\mathbf{R}$ in Scott's representation theorem can be viewed as a special case of this [Hay85]. In another direction, the category of smooth manifolds can be constructed as the Karoubi envelope of the category $\mathbf{C}$ of the open subsets of all Euclidean spaces, with smooth maps between them (see [Law89], page 267). For such classical results about the Karoubi envelope, it would be interesting to study how they hold up in univalent foundations for the different choices $\overline{\mathbf{C}}$ and $\hat{\mathbf{C}}$ of 'the Karoubi envelope'.

Bibliography

[ACU14] Thosten Altenkirch, James Chapman, and Tarmo Uustalu. "Monads need not be endofunctors". In: (2014). DOI: 10.2168/LMCS-11(1:3)2015. eprint: arXiv:1412.7148.

[AH76] K. Appel and W. Haken. "Every planar map is four colorable". In: *Bulletin of the American Mathematical Society* 82.5 (1976), pp. 711–712.

[AKS15] Benedikt Ahrens, Krzysztof Kapulkin, and Michael Shulman. "Univalent categories and the Rezk completion". In: *Mathematical Structures in Computer Science* 25.5 (Jan. 2015), pp. 1010–1039. ISSN: 1469-8072. DOI: 10.1017/s0960129514000486. URL: http://dx.doi.org/10.1017/S0960129514000486.

[AL17] Benedikt Ahrens and Peter LeFanu Lumsdaine. "Displayed Categories". In: (2017). DOI: 10.23638/LMCS-15(1:20)2019. eprint: arXiv:1705.04296.

[ARV10] J. Adámek, J. Rosický, and E. M. Vitale. *Algebraic Theories: A Categorical Introduction to General Algebra*. Cambridge Tracts in Mathematics. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511760754.

[AW23] Benedikt Ahrens and Kobe Wullaert. *Category Theory for Programming*. 2023. eprint: arXiv:2209.01259.

[Bez+20] Marc Bezem et al. *Construction of the Circle in UniMath*. 2020. arXiv: 1910.01856 [math.LO].

[Bor94] Francis Borceux. *Handbook of Categorical Algebra*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1994.

[Chu32] Alonzo Church. "A Set of Postulates for the Foundation of Logic". In: *Annals of Mathematics* 33.2 (1932), pp. 346–366. ISSN: 0003486X, 19398980. URL: http://www.jstor.org/stable/1968337 (visited on 09/25/2024).

[Chu36] Alonzo Church. "An Unsolvable Problem of Elementary Number Theory". In: *American Journal of Mathematics* 58.2 (1936), pp. 345–363. ISSN: 00029327, 10806377. URL: http://www.jstor.org/stable/2371045 (visited on 09/25/2024).

[Fou24] Python Software Foundation. Oct. 2024. URL: https://docs.python.org/3/reference/expressions.html#lambda.

[Fre72] Peter Freyd. "Aspects of topoi". In: *Bulletin of the Australian Mathematical Society* 7.1 (Aug. 1972), pp. 1–76. ISSN: 1755-1633. DOI: 10.1017/S0004972700044828. URL: http://dx.doi.org/10.1017/S0004972700044828.

[Gon08] Georges Gonthier. "The Four Colour Theorem: Engineering of a Formal Proof". In: *Computer Mathematics*. Ed. by Deepak Kapur. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 333–333. ISBN: 978-3-540-87827-8.

[HAL+17] THOMAS HALES et al. "A FORMAL PROOF OF THE KEPLER CONJECTURE". In: *Forum of Mathematics, Pi* 5 (2017), e2. DOI: 10.1017/fmp.2017.1.

[Hal02] Thomas C. Hales. *The Kepler conjecture*. 2002. arXiv: math/9811078 [math.MG]. URL: https://arxiv.org/abs/math/9811078.

[Hay85] Susumu Hayashi. "Adjunction of semifunctors: Categorical structures in nonextensional lambda calculus". In: *Theoretical Computer Science* 41 (1985), pp. 95–104. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(85)90062-3. URL: https://www.sciencedirect.com/science/article/pii/0304397585900623.

[HP89] J. Martin E. Hyland and Andrew M. Pitts. "The theory of constructions: Categorical semantics and topos-theoretic models". In: *Contemporary Mathematics* 92 (1989), pp. 137–199. DOI: 10.1090/conm/092/1003199. URL: http://dx.doi.org/10.1090/conm/092/1003199.

[HS93] Raymond Hoofman and Harold Schellinx. *Models of the untyped lambda-calculus in semi cartesian closed categories*. ILLC Prepublication Series for Mathematical Logic and Foundations, ML-93-05. Feb. 1993. eprint: https://eprints.illc.uva.nl/id/eprint/1329/1/ML-1993-05.text.pdf.

[Hur95] Antonius J. C. Hurkens. "A simplification of Girard's paradox". In: *Typed Lambda Calculi and Applications*. Ed. by Mariangiola Dezani-Ciancaglini and Gordon Plotkin. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 266–278. ISBN: 978-3-540-49178-1.

[Hyl14] Martin Hyland. "Towards a Notion of Lambda Monoid". In: *Electronic Notes in Theoretical Computer Science* 303 (2014). Proceedings of the Workshop on Algebra, Coalgebra and Topology (WACT 2013), pp. 59–77. ISSN: 1571-0661. DOI: https://doi.org/10.1016/j.entcs.2014.02.004. URL: https://www.sciencedirect.com/science/article/pii/S1571066114000309.

[Hyl17] J.M.E. Hyland. "Classical lambda calculus in modern dress". In: *Mathematical Structures in Computer Science* 27.5 (2017), pp. 762–781. DOI: 10.1017/S0960129515000377.

[Kam] Mark Kamsma. *Show that the Yoneda embedding preserves exponential objects*. Mathematics Stack Exchange. URL:https://math.stackexchange.com/q/3511278 (version: 2022-12-20). eprint: https://math.stackexchange.com/q/3511278. URL: https://math.stackexchange.com/q/3511278.

[KL18] Chris Kapulkin and Peter LeFanu Lumsdaine. *The Simplicial Model of Univalent Foundations (after Voevodsky)*. 2018. arXiv: 1211.2851 [math.LO]. URL: https://arxiv.org/abs/1211.2851.

[Kle36] S. C. Kleene. "λ-definability and recursiveness". In: *Duke Mathematical Journal* 2.2 (June 1936). ISSN: 0012-7094. DOI: 10.1215/s0012-7094-36-00227-2. URL: http://dx.doi.org/10.1215/s0012-7094-36-00227-2.

[KS06] Masaki Kashiwara and Pierre Schapira. *Categories and sheaves*. Vol. 332. Grundlehren der mathematischen Wissenschaften [Fundamental Principles of Mathematical Sciences]. Springer-Verlag, Berlin, 2006, pp. x+497. ISBN: 978-3-540-27949-5; 3-540-27949-0. DOI: 10.1007/3-540-27950-4. URL: https://doi.org/10.1007/3-540-27950-4.

[Law69] F. William Lawvere. "Diagonal arguments and cartesian closed categories". In: *Category Theory, Homology Theory and their Applications II*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1969, pp. 134–145. ISBN: 978-3-540-36101-5. DOI: https://doi.org/10.1007/BFb0080769.

[Law89] F. William Lawvere. "Qualitative distinctions between some toposes of generalized graphs". In: *Categories in computer science and logic (Boulder, CO, 1987)*. Vol. 92. Contemp. Math. Amer. Math. Soc., Providence, RI, 1989, pp. 261–299. ISBN: 0-8218-5100-4. DOI: 10.1090/conm/092/1003203. URL: https://doi.org/10.1090/conm/092/1003203.

[Mac98] Saunders Mac Lane. *Categories for the working mathematician*. Second. Vol. 5. Graduate Texts in Mathematics. Springer-Verlag, New York, 1998, pp. xii+314. ISBN: 0-387-98403-8.

[Mar71] Per Martin-Löf. "A theory of types". Preprint, Stockholm University. 1971.

[MM94] Saunders Mac Lane and Ieke Moerdijk. "Categories of Functors". In: *Sheaves in Geometry and Logic: A First Introduction to Topos Theory*. New York, NY: Springer New York, 1994, pp. 24–63. ISBN: 978-1-4612-0927-0. DOI: 10.1007/978-1-4612-0927-0_3. URL: https://doi.org/10.1007/978-1-4612-0927-0_3.

[MW03] Alexandre Miquel and Benjamin Werner. "The Not So Simple Proof-Irrelevant Model of CC". In: *Types for Proofs and Programs*. Ed. by Herman Geuvers and Freek Wiedijk. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 240–258. ISBN: 978-3-540-39185-2.

[nLa24a] nLab authors. *adjoint equivalence*. https://ncatlab.org/nlab/show/adjoint+equivalence. Revision 17. May 2024.

[nLa24b] nLab authors. *Grothendieck fibration*. https://ncatlab.org/nlab/show/Grothendieck+fibration. Revision 113. Feb. 2024.

[Ora22] Oracle. 2022. URL: https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html.

[Rie14] Emily Riehl. *Categorical Homotopy Theory*. New Mathematical Monographs. Cambridge University Press, 2014.

[Sco16] Dana S. Scott. *Greetings to the Participants at "Strachey 100"*. https://www.cs.ox.ac.uk/strachey100/Strachey_booklet.pdf. A talk read out at the Strachey 100 centenary conference. 2016.

[Sco72] Dana Scott. "Continuous lattices". In: *Toposes, Algebraic Geometry and Logic*. Ed. by F. W. Lawvere. Berlin, Heidelberg: Springer Berlin Heidelberg, 1972, pp. 97–136. ISBN: 978-3-540-37609-5. URL: http://dx.doi.org/10.1007/BFb0073967.

[SEL02] PETER SELINGER. "The lambda calculus is algebraic". In: *Journal of Functional Programming* 12.6 (2002), pp. 549–566. DOI: 10.1017/S0956796801004294.

[SH80] J. P. Seldin and J. R. Hindley, eds. *To H.B. Curry: Essays on Combinatory Logic and Formalism*. en. San Diego, CA: Academic Press, Sept. 1980.

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

[Tur37] A. M. Turing. "Computability and λ-Definability". In: *The Journal of Symbolic Logic* 2.4 (1937), pp. 153–163. ISSN: 00224812. URL: http://www.jstor.org/stable/2268280 (visited on 09/25/2024).

[Uni13] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study: https://homotopytypetheory.org/book, 2013.

[Voe14] Vladimir Voevodsky. *Experimental library of univalent formalization of mathematics*. 2014. arXiv: 1401.0053 [math.HO]. URL: https://arxiv.org/abs/1401.0053.

Appendix A

Alternative definitions

The literature, there are many different but equivalent definitions, carrying many different names, for the objects that called 'algebraic theories' in Section 4.1. Also, many of these different names are attached to differently defined objects in different sources. This section will showcase some of the various definitions.

In this section, we will denote the finite set $[\![n]\!] = \{1, \dots, n\}$.

A.1 Abstract Clone

Definition A.1. An algebraic theory as presented in Section 4.1, is usually called an *abstract clone*. In this thesis, outside of this specific section, we will call it algebraic theory to be consistent with the names that Hyland attaches to objects.

Remark A.2. The definition of algebraic theory that Hyland gives is closest to that of an abstract clone. However, instead of a sequence of sets $(T_n)_n$, he requires a functor $T : F \to \mathbf{Set}$ (with $F \subseteq \mathbf{FinSET}$ the skeleton category of finite sets $F_0 = \{[\![0]\!], [\![1]\!], \dots\}$), and he requires $\bullet : T_m \times T_n^m \to T_n$ to be dinatural in n and natural in m.

Using naturality, one can show that with such a functor we have $x_{n,i} = f(a)(x_{1,1})$ for the function $a(1) = i : [\![n]\!]$.

Alternatively, using the same naturality, one can show that this functor sends a morphism $a : [\![m]\!] \to [\![n]\!]$ to the function $T_m \to T_n$ given by

$$f \mapsto f \bullet (x_{n,a(i)})_i.$$

If we take this to be the definition of our functor on morphisms, the (di)naturality in m and n can be shown using the associativity and the laws about the interaction between $\bullet$ and the x_i.

Since any additional properties mean extra complexity when formalizing, and since the proofs rarely use the functor structure, we decided to reduce the functor $T : \mathbf{FinSET} \to \mathbf{Set}$ to a sequence of sets $(T_n)_n$.

A.2 Lawvere Theory

Definition A.3. An *algebraic theory* as presented in [ARV10] is a small category with finite products.

Definition A.4. An *algebra* for an algebraic theory T is a finite-product-preserving functor $T \to \mathbf{Set}$.

This definition is more general than the definition of algebraic theory in Section 4.1. To make it equivalent, we have to be more specific about the objects of the category:

Definition A.5. A *Lawvere theory*, or *one-sorted algebraic theory* is a category L, with $L_0 = \{0, 1, \dots\}$, such that $n = 1^n$, the n-fold product.

Lemma A.6 (One direction: `algebraic_theory_to_lawvere`)**.** *There is an equivalence between abstract clones and Lawvere theories.*

Proof. Let C be an abstract clone. We construct a Lawvere theory L as follows: We have objects $L_0 = \{0, 1, \dots\}$ and morphisms $L(m, n) = C_m^n$. The identity morphism is $\mathrm{id}_n = (x_i)_i : L(n, n)$ and for $f : L(l, m)$, $g : L(m, n)$, we have composition

$$f \cdot g = (g_i \bullet f)_i : L(l, n).$$

Lastly, we have product projections $\pi_{n,i} = x_{n,i} : L(n, 1)$ for all $1 \leqslant i \leqslant n$.

Conversely, if L is a Lawvere theory, we construct an abstract clone C as follows: We take $C_n = L(n, 1)$. We take the $x_{n,i}$ to be the product projections $\pi_i : L(n, 1)$ and we define the substitution $f \bullet g = \langle g_i \rangle_i \cdot f$ the composite of the product morphism $\langle g_i \rangle_i$ with f. □

A.2.1 Algebras for Lawvere Theories

Lemma A.7. *Let C be an abstract clone, and L be its associated Lawvere theory by the equivalence given above. A C-algebra is equivalent to an algebra for L.*

Proof. Let A be a C-algebra. We will construct a functor $F : L \to \mathbf{Set}$ as follows: We take $F(n) = A^n$. We define the action on morphisms as

$$F(f)(a) = (f_i \bullet a)_i$$

for $f : L(m, n) = L(m, 1)^n$ and $a : A^m$.

Conversely, let $F : L \to \mathbf{Set}$ be a functor. We take the C-algebra A, with $A = F(1)$ and for $f : C_n = L(n, 1)$ and $a : A^n$, $f \bullet a = F(f)(a)$. □

A.2.2 Presheaves for Lawvere Theories

Lemma A.8 (`algebraic_presheaf_weq_lawere_presheaf`)**.** *Let C be an abstract clone, and L be its associated Lawvere theory by the equivalence given above. A C-presheaf is equivalent to a presheaf over L.*

Proof. The correspondence between C-presheaves P and presheaves Q over L is as follows:

The sets P_n correspond to the action of Q on objects $Q(n)$.

The P-action $P_m \times C_n^m \to P_n$ corresponds to the action of Q on morphisms $L(n, m) \times Q(m) \to Q(n)$. □

A.3 Cartesian Operad

Definition A.9. A *cartesian operad* T is a functor $T : F \to \mathbf{Set}$, together with an 'identity' element $\mathrm{id} : T(1)$ and for all $m, n_1, \dots, n_m : \mathbb{N}$, a composition map

$$T(m) \times \prod_i T(n_i) \to T\left(\sum_i n_i\right),$$

written $(f, g_1, \dots, g_m) \mapsto f[g_1, \dots, g_m]$, satisfying some identity, associativity and naturality conditions (see [Hyl14], Definition 2.1, for more details).

Remark A.10. One can arrive at this concept as a standalone definition, or one can view a cartesian operad as a *cartesian multicategory* with one element. A multicategory is a category in which morphisms have type $C((X_1, X_2, \ldots, X_n), Y)$ instead of $C(X, Y)$. A cartesian multicategory is a multicategory in which one can permute the X_i of a morphism and has 'contraction' and 'weakening' operations:

$$C((X_1, \ldots, X_i, X_i, \ldots, X_n), Y) \to C((X_1, \ldots, X_i, \ldots, X_n), Y),$$
$$C((X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n), Y) \to C((X_1, \ldots, X_{i-1}, X_i, X_{i+1}, \ldots, X_n), Y).$$

Lemma A.11. *There is an equivalence between abstract clones and cartesian operads.*

Proof. Let C be an abstract clone. We define a cartesian operad T with $T(n) = C_n$ and $T(f)(t) = t \bullet (x_{f(1)}, \ldots, x_{f(m)})$ for $f : [\![m]\!] \to [\![n]\!]$ and $t : C_m$. The identity element is $x_{1,1}$ and the composition $f[g_1, \ldots, g_n]$, for $f : C_m$ and $g_i : C_{n_i}$, is given by lifting all terms to $C_{\sum_i n_i}$ and then substituting:

$$f[g_1, \ldots, g_n] = f \bullet (T(\iota_i)(g_i))$$

for $\iota_i : [\![n_i]\!] \hookrightarrow [\![\sum_i n_i]\!]$ the pairwise disjoint injections.

Conversely, given a cartesian operad T, we construct an abstract clone C with $C_n = T([\![n]\!])$. The variables are $x_{n,i} = \iota_{n,i}(\mathrm{id})$, for $\iota_{n,i} : [\![1]\!] \hookrightarrow [\![n]\!]$ the morphism that sends 1 to i. The substitution $f \bullet (g_i)_i$ for $g_1, \ldots, g_m : T(n)$ is given by composing and then identifying some variables:

$$f \bullet (g_i)_i = T(\pi)(f[g_1, \ldots, g_m])$$

for $\pi : [\![mn]\!] \to [\![n]\!]$ the function that sends $i + 1$ to $(i \mod n) + 1$. □

A.4 Relative Monad

Definition A.12. Let $S : C \to D$ be a functor. A *relative monad* on S is a functor $T : C \to D$, together with a natural transformation $\eta : S \Rightarrow T$ and a 'kleisli extension' $(-)^* : D(S(X), T(Y)) \to D(T(X), T(Y))$, natural in both S and T, such that for all $f : D(S(X), T(Y))$ and $g : D(S(Y), T(Z))$,

$$\eta_X^* = \mathrm{id}_{TX}, \qquad f = \eta_X \cdot f^* \qquad \text{and} \qquad (f \cdot g^*)^* = f^* \cdot g^*.$$

Remark A.13. Note that for an adjunction $F \dashv G$, $F \bullet G$ gives a monad. In the same way, there exists a notion of *relative adjunction*, from which we can obtain a relative monad (see [ACU14], Theorem 2.10).

Remark A.14. Now, there is a result that states: There is an equivalence between abstract clones and relative monads on the embedding $\iota :$ **FinSET** $\hookrightarrow$ **Set**.

Note that the objects of **FinSET** are defined to be sets X, together with a proof that there exists some $n : \mathbb{N}$ and some bijection $f : X \xrightarrow{\sim} [\![n]\!]$. This existence of n and f is given by the propositional truncation $\left\| \sum_{n:\mathbb{N}}, X \xrightarrow{\sim} [\![n]\!] \right\|$.

Classically, this construction starts with "fix, for all $X :$ **FinSET**, a bijection $f : X \xrightarrow{\sim} [\![n]\!]$". However, since X only provides *mere existence* of such a bijection without choosing one (by the propositional truncation), there is no way to obtain f without using the axiom of choice.

Now, we can partially circumvent this problem by noting that we have a fully faithful and essentially surjective embedding $F \to$ **FinSET** (for F a skeleton of finite sets), which induces an adjoint equivalence $[F,$ **Set**$] \xrightarrow{\sim} [$**FinSET**, **Set**$]$, so we can lift the functor part of a relative monad on $F \to$ **Set** to **FinSET** $\to$ **Set**. We can also lift the natural transformation using this equivalence. However, the kleisli extension cannot be lifted using this and we are stuck.

Therefore, we will prove a modified statement:

Lemma A.15. *There is an equivalence between abstract clones and relative monads on the embedding* $\iota : F \hookrightarrow \mathbf{Set}$*.*

Proof. Let C be an abstract clone. We define a relative monad T as follows: We take $T([\![n]\!]) = C_n$. For a morphism $a : F([\![m]\!], [\![n]\!])$. We take $T(a)(f) = f \bullet (x_{a(i)})_i$. We define $\eta_{[\![n]\!]}(i) = x_{n,i}$. Finally, for $g : \mathbf{Set}([\![m]\!], T([\![n]\!]))$, we define $g^*(f) = f \bullet g$.

Conversely, let $(T, \eta, (\cdot)^*)$ be a relative monad on the embedding $\iota : F \hookrightarrow \mathbf{Set}$. We define an abstract clone C with $C_n = T([\![n]\!])$. Substitution is defined as $f \bullet g = g^*(f)$ for $f : C_m$ and $g : C_n^m$ and we have variables $x_{n,i} = \eta_{[\![n]\!]}(i)$. □

A.5 Monoid in a Skew-Monoidal Category

For details and a general treatment, see [ACU14], Section 3 and specifically Theorem 3.4.

Definition A.16. Consider the category $[F, \mathbf{Set}]$. Note that we have a functor $\iota : F \hookrightarrow \mathbf{Set}$ and that **Set** has colimits. Therefore, we can define a 'tensor product' on functors $[F, \mathbf{Set}]$ as

$$F \otimes G = G \bullet \mathrm{Lan}_\iota F.$$

Together with the 'unit' ι, this gives $[F, \mathbf{Set}]$ a 'skew-monoidal category' structure.

Definition A.17. Given a (skew-)monoidal category $(C, \otimes, I)$, a *monoid* in this category is an object $T : C$, together with a 'multiplication' $\mu : C(T \otimes T, T)$ and a 'unit' morphism $\eta : C(I, T)$ satisfying a couple of laws (see [Mac98], Section III.6).

Lemma A.18. *There exists an equivalence between relative monads on* $\iota : F \hookrightarrow \mathbf{Set}$ *and monoids in the skew-monoidal category* $[F, \mathbf{Set}]$.

Proof. A monoid in $[F, \mathbf{Set}]$ consists of an object $T : [F, \mathbf{Set}]$, together with natural transformations $\mu : T \otimes T \Rightarrow T$ and $\eta : \iota \Longrightarrow T$. We can immediately see the functor T and natural transformation η of the relative monad pop up here. The kleisli extension corresponds to μ; they are related to each other via the properties of the left Kan extension of T. □

Appendix B

Weak Cartesian Closed Categories

In Definition 4.51, we defined the endomorphism λ-theory of a reflexive object in a cartesian closed category. However, as it turns out, we do not need all of the properties of a cartesian closed category. It suffices to consider a *weak* cartesian closed category. In fact, even a 'semi cartesian closed category' will suffice, but in this section, we will restrict ourselves to a weak cartesian closed category, as this notion suits our purposes well enough.

It seems that semi cartesian closed categories were first defined in [Hay85] and then [HS93] gave a slightly different definition, as well as the very related definition of a weak cartesian closed category. The latter paper also contains the construction of the endomorphism λ-theory.

Definition B.1. Let C be a category with binary products (we will call the projections π_1 and π_2), and take $A, B : C$. A *semi-exponential object* is an object $A^B : C$, with a morphism $\mathrm{ev} : C(A^B \times B \to A)$ and a function

$$\mathrm{cur}_X : C(X \times B \to A) \to C(X, A^B)$$

for all X, such that

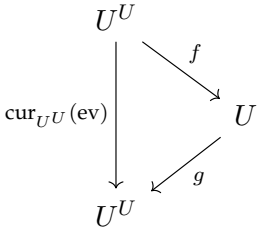
$$\langle g \cdot \mathrm{cur}_X(f), h\rangle \cdot \mathrm{ev} = \langle g, h\rangle \cdot f \quad \text{and} \quad \mathrm{cur}_Y((g \times \mathrm{id}_B) \cdot f) = g \cdot \mathrm{cur}_X(f)$$

for all $f : C(X \times B, A)$, $g : C(Y, X)$ and $h : C(Y, B)$.

Definition B.2. A *weak cartesian closed category* is a category C with a terminal object and binary products, together with a choice for a semi-exponential object A^B for all $A, B : C$.

Remark B.3. Since the requirements for semi-exponential objects do not involve the usual universal properties, a weak cartesian closed structure (or a semi-exponential object) is not unique up to isomorphism. For example, see example 2.1 and 2.2 in [HS93] for two distinct weak cartesian closed structures on a category.

Definition B.4. A *reflexive object* in a category C with binary products is an object U with an exponential U^U and morphisms f and g, forming the following commutative diagram:



Definition B.5. If we have a reflexive object (U, f, g) in a category C with finite products, we can endow the endomorphism algebraic theory $E_C(U)$ (Definition 4.51) with a λ-theory structure with β-equality, with

$$\rho(s) = ((s \cdot g) \times \mathrm{id}_U) \cdot \mathrm{ev} : C(U^{n+1}, U)$$

and

$$\lambda(t) = \mathrm{cur}_{U^n}(t) \cdot f : C(U^n, U)$$

for $s : E(U)_n$ and $t : E(U)_{n+1}$.

Lemma B.6. *The definition above indeed yields a λ-theory with β-equality.*

Proof. Note that for morphisms $f : C(U^{m+1}, U)$ and $g : C(U^n, U)^m$,

$$(\langle g_i \rangle_i \times \mathrm{id}_U) \cdot f = f \bullet (\iota_{n,1}(g_1), \ldots, \iota_{n,1}(g_m), x_{n+1}).$$

Then

$$\begin{aligned}
\rho(s \bullet t) &= ((\langle t_i \rangle_i \cdot s \cdot g) \times \mathrm{id}_U) \cdot \mathrm{ev} \\
&= (\langle t_i \rangle_i \times \mathrm{id}_U) \cdot ((s \cdot g) \times \mathrm{id}_U) \cdot \mathrm{ev} \\
&= \rho(s) \bullet ((\iota_{n,1}(t_i))_i + (x_{n+1})),
\end{aligned}$$

$$\begin{aligned}
\lambda(s) \bullet t &= \langle t_i \rangle_i \cdot \mathrm{cur}(t) \cdot f \\
&= \mathrm{cur}_{U^n}((\langle t_i \rangle_i \times \mathrm{id}_U) \cdot t) \cdot f \\
&= \lambda(s \bullet ((\iota_{n,1}(t_i))_i + (x_{n+1}))),
\end{aligned}$$

where we use the property of cur, and

$$\begin{aligned}
\rho(\lambda(s)) &= \langle \pi_1 \cdot \mathrm{cur}_{U^n}(t) \cdot \mathrm{cur}_{U^U}(\mathrm{ev}), \pi_2 \rangle \cdot \mathrm{ev} \\
&= \langle \pi_1 \cdot \mathrm{cur}_{U^n}(t), \pi_2 \rangle \cdot \mathrm{ev} \\
&= t
\end{aligned}$$

where we use the property of ev twice. □

We get the following as a special case, in which U^U is not only a retract of U, but actually equal to U.

Corollary B.7. *Let L be an algebraic theory. Suppose that its associated Lawvere theory C is a weak cartesian closed category and suppose that for $1 : C$, we have $1^1 = 1$. Then we can give L a λ-theory structure with β-equality, given by*

$$\lambda_n(s) = \mathrm{cur}_n(s) \quad \textit{and} \quad \rho(t) = \mathrm{ev} \bullet ((\iota_{n,1}(t_i))_i + (x_{n+1})).$$

Now, we will show that the semi-exponential objects satisfy some properties that justify the use of the exponential notation. These properties allow us to construct semi-exponentials on products from the semi-exponentials of their components.

Remark B.8. Note that in a category C with a terminal object I, we have exponential objects

$$A^I = A \quad \text{and} \quad I^A = I$$

for all objects A. Therefore, we get semi-exponential objects for free if we have a terminal object.

Lemma B.9. *Take $A, A', B : C$. If A^B and ${A'}^B$ are semi-exponential objects, then $A^B \times (A')^B$ is the semi-exponential object $(A \times A')^B$.*

Proof. We have

$$\begin{aligned} \mathrm{ev} : C(A^B \times B \to A) \quad \text{and} \quad \mathrm{ev}' : C({A'}^B \times B \to A'), \\ \mathrm{cur}_X : C(X \times B, A) \to C(X, A^B) \quad \text{and} \quad \mathrm{cur}'_X : C(X \times B, A') \to C(X, {A'}^B). \end{aligned}$$

Define

$$\mathrm{ev}'' = \langle\langle \pi_1 \cdot \pi_1, \pi_2 \rangle \cdot \mathrm{ev}, \langle \pi_1 \cdot \pi_2, \pi_2 \rangle \cdot \mathrm{ev} \rangle : C(A^B \times {A'}^B, A \times A')$$

and for $f : C(X \times B, A \times A')$,

$$\mathrm{cur}''_X(f) = \langle \mathrm{cur}_X(f \cdot \pi_1), \mathrm{cur}_X(f \cdot \pi_2) \rangle : C(X, A^B \times A^{B'}).$$

These satisfy the required equations. □

Lemma B.10. *Take $A, B, B' : C$. If A^B and $(A^B)^{B'}$ are semi-exponential objects, then $(A^B)^{B'}$ is the semi-exponential object $A^{B \times B'}$.*

Proof. We have

$$\begin{aligned} \mathrm{ev} : C(A^B \times B \to A) \quad \text{and} \quad \mathrm{ev}' :: C((A^B)^{B'} \times B' \to A^B), \\ \mathrm{cur}_X : C(X \times B, A) \to C(X, A^B) \quad \text{and} \quad \mathrm{cur}'_X : C(X \times B', A^B) \to C(X, (A^B)^{B'}). \end{aligned}$$

Define

$$\mathrm{ev}'' = \langle (\mathrm{id}_{(A^B)^{B'}} \times \pi_2) \cdot \mathrm{ev}', \pi_2 \cdot \pi_1 \rangle \cdot \mathrm{ev} : C((A^B)^{B'} \times (B \times B'), A)$$

and for $f : C(X \times (B \times B'), A)$,

$$\mathrm{cur}''_X(f) = \mathrm{cur}'_X(\mathrm{cur}_{X \times B'}(\langle \pi_1 \cdot \pi_1, \langle \pi_2, \pi_1 \cdot \pi_2 \rangle \rangle \cdot f)) : C(X, (A^B)^{B'}).$$

Then, for $f : C(X \times (B \times B'), A)$, $g : C(Y, X)$ and $h : C(Y, B \times B')$,

$$\begin{aligned} \langle g \cdot \mathrm{cur}''_X(f), h \rangle \cdot \mathrm{ev}'' &= \langle\langle g \cdot \mathrm{cur}'_X(\mathrm{cur}_{X \times B'}(\langle \pi_1 \cdot \pi_1, \langle \pi_2, \pi_1 \cdot \pi_2 \rangle \rangle \cdot f)), h \cdot \pi_2 \rangle \cdot \mathrm{ev}', h \cdot \pi_1 \rangle \cdot \mathrm{ev} \\ &= \langle\langle\langle g, h \cdot \pi_2 \rangle, h \cdot \pi_1 \rangle \cdot \pi_1 \cdot \pi_1, \langle\langle\langle g, h \cdot \pi_2 \rangle, h \cdot \pi_1 \rangle \cdot \pi_2, \langle\langle g, h \cdot \pi_2 \rangle, h \cdot \pi_1 \rangle \cdot \pi_1 \cdot \pi_2 \rangle\rangle \cdot f \\ &= \langle g, h \rangle \cdot f \end{aligned}$$

and

$$\begin{aligned} \mathrm{cur}''_Y((g \times \mathrm{id}_B) \cdot f) &= \mathrm{cur}'_Y(\mathrm{cur}_{Y \times B'}(((g \times \mathrm{id}_{B'}) \times \mathrm{id}_B) \cdot \langle \pi_1 \cdot \pi_1, \langle \pi_2, \pi_1 \cdot \pi_2 \rangle \rangle \cdot f)) \\ &= \mathrm{cur}'_Y((g \times \mathrm{id}_{B'}) \cdot \mathrm{cur}_{X \times B'}(\langle \pi_1 \cdot \pi_1, \langle \pi_2, \pi_1 \cdot \pi_2 \rangle \rangle \cdot f)) \\ &= g \cdot \mathrm{cur}'_X(\mathrm{cur}_{X \times B'}(\langle \pi_1 \cdot \pi_1, \langle \pi_2, \pi_1 \cdot \pi_2 \rangle \rangle \cdot f)), \end{aligned}$$

which concludes the proof. □

Now, we can show that giving an algebraic theory a λ-theory structure gives its corresponding Lawvere theory (Lemma A.6) a weak cartesian closed structure.

Lemma B.11. *Let L be a λ-theory with β-equality. Its corresponding Lawvere theory C is a weak cartesian closed category.*

Proof. Note that by definition of a Lawvere theory, every $n : C$ is the n-fold product of 1. Therefore, binary products are given by addition on the natural numbers, and 0 is the terminal object.

Now, we define $1^1 = 1$, with

$$\mathrm{ev} = \rho(x_1) : C(2,1) \quad \text{and} \quad \mathrm{cur}_n(f) = \lambda(f) : C(n,1)$$

for $f : C(n+1,1)$. The following shows that this is indeed a semi-exponential object:

$$\rho(x_1) \bullet (\lambda(f) \bullet g, h) = \rho(x_1 \bullet \lambda(f)) \bullet (g + (h)) = f \bullet (g + (h))$$

and

$$\lambda(f \bullet ((\iota_{n,1}(g_i))_i + (x_{n+1,n+1}))) = \lambda(f) \bullet g$$

Now, by the remark and the lemmas above, we have for $m, n : C$ (both nonzero),

$$m^n = (1 \times \cdots \times 1)^n = 1^n \times \cdots \times 1^n$$

and

$$1^n = 1^{1 \times \cdots \times 1} = ((1^1)^1 \dots)^1 = 1$$

so C has semi-exponential objects and it is a weak cartesian closed category. □

Remark B.12. Note that if L has both β- and η-equality, then for 1^1,

$$\mathrm{cur} : C(n+1,1) \to C(n,1),$$

which is given by λ, has an inverse given by ρ. Furthermore, since λ and ρ respect substitution, these constitute a natural isomorphism between $C(- + 1, 1)$ and $C(-, 1)$, so the semi-exponential object $1^1 = 1$ is in fact an exponential object. Then we have for all m and n, an exponential object

$$m^n = (1 \times \cdots \times 1)^{1 \times \cdots \times 1} = 1^{1 \times \cdots \times 1} \times \cdots \times 1^{1 \times \cdots \times 1} = 1 \times \cdots \times 1 = m,$$

so C is cartesian closed.

Index