Higher Inductive Types Via Impredicative Encodings

MSc Thesis *(Afstudeerscriptie)*

written by

Stefano Volpe

under the supervision of **Dr Benno van den Berg**, and submitted to the Examinations Board in partial fulfilment of the requirements for the degree of

MSc in Logic

at the *Universiteit van Amsterdam*.

Date of the public defence:	**Members of the Thesis Committee:**
August 27, 2025	Dr Benno van den Berg
	Dr Malvin Gattinger
	Prof Dr Herman Geuvers
	Dr Kristina Sojakova

Institute for Logic, Language and Computation

It would be so nice if something made sense for a change.
— Alice, from *Alice's Adventures in Wonderland*
by Lewis Carroll

Abstract

In dependent type theory with an impredicative universe, the usual encodings for inductive types borrowed from System F do not satisfy the dependent elimination rule (equivalently, no form of the η-rule holds). Two refinements of these encodings have been proposed to remedy this. Both apply to all $\mathcal{W}$-types. The first is by S. Awodey, J. Frey, and S. Speight [4]. They manage to recover dependent elimination into 0-types of the impredicative universe by assuming the existence of Σ-types, identity types, and function extensionality. By also assuming the existence of a natural numbers type, X. Ripoll Echeveste [21], inspired by the ideas of M. Shulman [23], devises a second, distinct refinement for $\mathcal{W}$-types. This time, dependent elimination is possible into the entire impredicative universe.

In this thesis, we show how both encodings can be extended to higher inductive types. While $\mathcal{W}$-types are a popular choice for a working definition of "inductive types", there are multiple competing definitions of "higher inductive types" in the literature. We test our generalisation of the first refinement on the class of higher inductive types defined by H. Basold, H. Geuvers, and N. van der Weide [7], and our generalisation of the second on the $\mathcal{W}$-suspensions by K. Sojakova [24]. Finally, we fully formalise the first refined encoding of higher inductive types in the Agda proof assistant [1].

Acknowledgements

Benno van den Berg, my supervisor, has had the utmost patience when dealing with me during my two years at the Master of Logic, first as a lecturer, and then during my thesis semester. Similarly, my "unofficial" supervisor, Daniël Otten, had also acted as a great teaching assistant before my thesis. It takes a great deal of energy to teach a computer scientist some mathematical logic! For this, I am very grateful to both. Similarly, I also wish to thank the rest of the committee for the interest they showed in my work. It was a pleasure to discuss it with them. My family surely deserves a mention, too. Everything Ilaria, mom, and dad could have done to make this adventure work, they did. I am relieved to see their efforts brought me here: the Master of Logic is a special place. If you have come across it, you know. There are too many students, from inside and outside the MoL room, for me to thank them all. Please allow me to single out Lu(crezia), Klarise, Emma, Bas, *Yoshi* (Josje), and Alyssa. They have never given up on me.

Having to say goodbye to all these people makes leaving Amsterdam a painful experience. Wherever I go, whomever I meet, I hope the good they have sowed in me is nurtured and shared.

Contents

Listings

Chapter 1.

Background

This thesis is on impredicative encodings of higher inductive types. As such, we assume very basic knowledge in category theory, and in particular functors and limits. An introductory account of the discipline is provided by T. Leinster [16]. Similarly, we expect the reader to be familiar with some foundational notions from homotopy type theory, namely function extensionality and homotopy levels. E. Rijke [20] offers a good text for beginners on this regard. Knowledge of proof assistants is only required when inspecting the Agda [1] formalisation of our work. The parts relevant to the text currently being read can be consulted by clicking on the cogwheel icon (⚙) accompanying our definitions, statements, examples, and listings. Some links actually point to the code from the Cubical Agda Library [3], on which our work depends. In case of network problems, the repository is also available as a compressed archive embedded within this PDF document.

All other background material necessary to understand our work is covered in this chapter. In particular, we introduce our working definition(s) of higher inductive types in Section 1.1, and the general idea behind impredicative encodings in Section 1.2. We only get into the details of the type theory we are working with in Section 1.3. Finally, our contributions are described in Section 1.4.

1.1. Higher Inductive Types

In homotopy type theory, higher inductive types (HITs) extend the notion of inductive types by allowing for *path (or "higher") constructors*. These generate paths for the type being defined. To avoid confusion, we refer to traditional constructors as *point constructors*. If an inductive type is freely generated by some collection of point constructors, a higher inductive type is freely generated by some collection of point and path constructors[1]. To date, there is no consensus on a single, general definition for HIT schemas. In this thesis, we test our encoding techniques on two different definitions:

1. Van der Weide's HITs as defined in H. Basold, H. Geuvers, and N. van der Weide [7][2]. We use them to generalise the set-truncated impredicative encodings by S. Awodey, J. Frey, and S. Speight [4];
2. $\mathcal{W}$-suspensions by K. Sojakova [24]. We use them in the generalisation of the impredicative encodings by X. Ripoll Echeveste [21].

Both definitions are described in the following.

[1] Throughout this work, all unqualified uses of the word "constructor" refer to point and path constructors collectively.

[2] This definition has seen adaptations in subsequent papers also by Van der Weide, so our appellation choice is for lack of a better name.

1.1.1. Van der Weide's Higher Inductive Types

These schemas will be our working definitions when giving impredicative encodings of set-truncated higher inductive types. Hence, we deviate from the original definition in that we assume the constants in the polynomial type constructors to belong to some fixed homotopy level. As per usual, polynomial type constructors are used as a grammar for constructor argument types. We use F, G, H… to refer to them. For the rest of this section, we fix a universe level ℓ and a homotopy level h.

Definition 1.1. ⚙ The type PolyTypeConstr $\ell\, h$ of *polynomial type constructors of universe levels ℓ and homotopy level h*, or simply "polynomials of level h", is defined as

$$\boldsymbol{F} :\equiv \text{Const } \boldsymbol{A} \mid \mathbb{X} \mid \boldsymbol{F} \otimes \boldsymbol{F} \mid \boldsymbol{F} \oplus \boldsymbol{F}$$

where $\boldsymbol{A} : h\text{–Type}_\ell :\equiv \sum_{X:\mathcal{U}_\ell} \text{is–}h\text{–Type } X$.

By substituting an actual h-type B for the 0-ary constructor $\mathbb{X}$ in a polynomial F and interpreting $\otimes$ ($\oplus$) into $\times$ ($+$), we can evaluate such polynomial at B. We write this as "$F[B]$", where the binary $\cdot\,[\cdot]$ operator has higher precedence than function application, so $fF[B]$ is to be read as $f(F[B])$, and not as $(fF)[B]$. In practice, we are only ever interested in homotopy sets (level 0) and higher homotopy levels, as mere propositions are not closed under finite sums, so we need to work with $S(Sh)$, where S is the "successor" constructor for homotopy levels.

Definition 1.2. ⚙ Let F be a polynomial type constructor of universe level ℓ and homotopy level $S(Sh)$. We define the following *evaluation map on types for F* recursively, where isOfHLevel$\times$ $(S(Sh))$ and isOfHLevel+ h witness $\text{is–}(S(Sh))\text{–Type}_\ell$ being closed under binary products and sums respectively.

$$\begin{aligned}
F[\cdot] &: (S(Sh))\text{–Type}_\ell \to (S(Sh))\text{–Type}_\ell \\
(\text{Const } A)[B] &:\equiv A \\
\mathbb{X}[B] &:\equiv B \\
(G \otimes H)[B] &:\equiv (\text{pr}_1 G[B] \times \text{pr}_1 H[B], \text{isOfHLevel}\times\ (S(Sh))(\text{pr}_2 G[B])(\text{pr}_2 H[B])) \\
(G \oplus H)[B] &:\equiv (\text{pr}_1 G[B] + \text{pr}_1 H[B], \text{isOfHLevel+}\ h(\text{pr}_2 G[B])(\text{pr}_2 H[B]))
\end{aligned}$$

Similarly, we can lift a function $f : B \to C$ between types of some homotopy level to a polynomial F whose constants belong to the same level (write "$F[[f]]$", where again $\cdot\,[[\cdot]]$ has higher precedence than function application).

Definition 1.3. ⚙ Let F be a polynomial type constructor of universe level ℓ and homotopy level $S(Sh)$. Let $(B,k),(C,l) : (S(Sh))\text{–Type}$. We define the following *evaluation map on functions from (B,k) to (C,l) for F* recursively.

$$\begin{aligned}
F[[\cdot]] &: (B \to C) \to \text{pr}_1 F[(B,k)] \to \text{pr}_1 F[(C,l)] \\
(\text{Const } (A,m))[[f]] &:\equiv \text{id}_A \\
\mathbb{X}[[f]] &:\equiv f \\
(G \otimes H)[[f]] &:\equiv \text{map}\times\ G[[f]]H[[f]] \\
(G \oplus H)[[f]] &:\equiv \text{map+}\ G[[f]]H[[f]]
\end{aligned}$$

At the set level (i.e., for $h \equiv -2$ and hence $S(Sh) \equiv 0$), which will be our setting, this extends to a functor on Set_ℓ, the precategory (i.e., non-univalent category with set-truncated hom-types) of homotopy sets in $\mathcal{U}_\ell$ and functions between them, in the obvious way. While we could generalise this observation to higher homotopy levels, we would need to lift the restriction of hom-types between any two objects being set-truncated, and this would result in working with wild[3] precategories (P. Capriotti and N. Kraus [10]), rather than precategories. As this generalisation is not necessary, we just stick to the set level when working with endofunctors in this subsection.

Proposition 1.4. ⚙ Let F be a polynomial type constructor of universe level ℓ and homotopy level 0. Then $F[\cdot]$ and $F[[\cdot]]$ are the actions on objects and maps respectively of an endofunctor on Set_ℓ that we denote by $\mathrm{polyFuncOnSet}\ F$.

Proof. The identity and composition conditions for functors hold definitionally if F is a constant or variable $\mathbb{X}$. For $F \equiv G \otimes H$, $(A, k) : 0\text{–}\mathrm{Type}_\ell$, we have

$$\begin{aligned}(G \otimes H)[\mathrm{id}_A] &\equiv \mathrm{map}\times\ G[[\mathrm{id}_A]]H[[\mathrm{id}_A]] \\ &\overset{\text{IH}}{=} \mathrm{map}\times\ \mathrm{id}_{\mathrm{pr}_1 G[(A,k)]}\mathrm{id}_{\mathrm{pr}_1 H[(A,k)]} \\ &\equiv \mathrm{id}_{\mathrm{pr}_1(G[(A,k)]\times H[(A,k)])} \\ &\equiv \mathrm{id}_{\mathrm{pr}_1(G\otimes H)[(A,k)]}\end{aligned}$$

and

$$\begin{aligned}(G \otimes H)[[g\circ f]] &\equiv \mathrm{map}\times\ G[[g\circ f]]H[[g\circ f]] \\ &\overset{\text{IH}}{=} \mathrm{map}\times\ (G[[g]]\circ G[[f]])(H[[g]]\circ H[[f]]) \\ &\equiv \mathrm{map}\times\ G[[g]]H[[g]]\circ\ \mathrm{map}\times\ G[[f]]H[[f]] \\ &\equiv (G \otimes H)[[g]]\circ(G \otimes H)[[f]].\end{aligned}$$

The case where $F \equiv G \oplus H$ is quite similar. □

Now, initial semantics for inductive types usually deals with all (point) constructors at the same time using a single endofunctor. We will do the same: consider a family $(H_i)_{i:\ \mathrm{Fin}\ k}$, of $k : \mathbb{N}$ polynomials of universe level ℓ and homotopy level 0. Each of its members induces an endofunctor on Set_ℓ as constructed in Proposition 1.4. We can easily define a new endofunctor $\bigoplus H$ on Set_ℓ by pointwise taking the k-ary coproduct as the action on objects. Such evaluation map can always be converted back to a family of k separate functions, one for each point constructor. This can be thought of as deconstructing an universal arrow for a k-ary coproduct diagram.

Definition 1.5. ⚙ Let $B : h\text{–}\mathrm{Type}_\ell$. Given a family $(H_i)_{i:\ \mathrm{Fin}\ k}$ of $k : \mathbb{N}$ polynomials of universe level ℓ and homotopy level h, we define a helper function

$$\mathrm{scatter} : \left(\mathrm{pr}_1\left(\bigoplus H\right)B \to \mathrm{pr}_1 B\right) \to \prod_{i:\ \mathrm{Fin}\ k} \mathrm{pr}_1 H_i[B] \to \mathrm{pr}_1 B$$

$$\mathrm{scatter}\ \alpha\ i :\equiv \alpha \circ \mathrm{in}_i$$

where in_i is the i-th constructor of k-ary sums.

[3]Recall that "wild" stands for "whose type of morphisms between any two fixed objects needs not be a homotopy set".

The inverse operation, which we will call cluster, will also be helpful: it is simply the map function for k-ary sums (i.e., the function computing the universal arrow for k-ary coproducts), with signature

$$\left(\prod_{i:\,\mathrm{Fin}\,k} \mathrm{pr}_1 H_i[B] \to \mathrm{pr}_1 B\right) \to \mathrm{pr}_1\Big(\bigoplus H\Big)[B] \to \mathrm{pr}_1 B.$$

Of course, these two functions are each other's inverses. Covered polynomial type constructors, we move to endpoints of path constructors (for which we use symbols t, r, s, …). These also follow a specific grammar.

Definition 1.6. ⚙ Let $(H_i)_{i:\,\mathrm{Fin}\,k}$ be a finite family[4] of $k : \mathbb{N}$ of polynomials of universe level ℓ and homotopy level h. Let F be a polynomial of universe level ℓ and homotopy level h. The type family

$$\text{PathConstructorTerm } HF : \text{PolyTypeConstr } \ell\, h \to \mathcal{U}_{S\ell}$$

is inductively defined by the following introduction rules. We read type PathConstructorTerm HFG as *"path constructor term over H from F to G"*.

$$\frac{\vdash t : A}{\vdash \text{ConstTerm } t : \text{PathConstructorTerm } HFA} \qquad \frac{}{\vdash \mathbb{x} : \text{PathConstructorTerm } HFF}$$

$$\frac{\vdash s : \text{PathConstructorTerm } HFH_i}{\vdash c_i s : \text{PathConstructorTerm } HF\mathbb{X}} \qquad \frac{\vdash s : \text{PathConstructorTerm } HF(G_1 \otimes G_2)}{\vdash \pi_j s : \text{PathConstructorTerm } HFG_j}\, j \in \{1,2\}$$

$$\frac{\vdash s_1 : \text{PathConstructorTerm } HFG_1 \quad \vdash s_2 : \text{PathConstructorTerm } HFG_2}{\vdash (s_1, s_2) : \text{PathConstructorTerm } HF(G_1 \otimes G_2)}$$

$$\frac{\vdash s : \text{PathConstructorTerm } HFG_j}{\vdash \mathrm{in}_j s : \text{PathConstructorTerm } HF(G_1 \oplus G_2)}\, j \in \{1,2\}$$

Much like for polynomials, we can also evaluate constructor terms for a type $B : h\text{–}\mathrm{Type}_\ell$. This time, however, we also need to specify that we are interpreting our k point constructors using some evaluation map $\mathrm{pr}_1((\bigoplus H)B) \to \mathrm{pr}_1 B$. We also need some constant $y : \mathrm{pr}_1 F[B]$ to evaluate the path argument x, so we write $r\ \ 【B, d, y】\ $. We adopt the convention that $fr\ \ 【B, d_i, y】\ $ is to be read as $f\big(r\ \ 【B, d_i, y】\ \ \big)$, rather than $(fr)\ \ 【B, d_i, y】\ $.

Definition 1.7. ⚙ Let $(H_i)_{i:\,\mathrm{Fin}\,k}$ $k : \mathbb{N}$ be a family of polynomials of universe level ℓ and homotopy level 0. Let $F, G : \text{PolyTypeConstr } \ell h$, $r : \text{PathConstructorTerm } HFG$, $B : h\text{–}\mathrm{Type}_\ell$, $d : \mathrm{pr}_1((\bigoplus H)B) \to \mathrm{pr}_1 B$, $y : \mathrm{pr}_1 F[B]$. We define $r\ \ 【B, d, y】\ $, the *value of path constructor term r at B-d-y*, via structural recursion on r.

[4] We use $1, 2, ..., k$ to denote the terms of type Fin k.

$$
\begin{aligned}
r \quad \llbracket B, d, y \rrbracket \quad &: \mathrm{pr}_1 G[B] \\
(\text{ConstTerm } a) \quad \llbracket B, d, y \rrbracket \quad &:\equiv a \\
\aleph \quad \llbracket B, d, y \rrbracket \quad &:\equiv y \\
(c_i s) \quad \llbracket B, d, y \rrbracket \quad &:\equiv \text{scatter } d\, i \left(s \quad \llbracket B, d, y \rrbracket \quad\right) \\
(\pi_1 s) \quad \llbracket B, d, y \rrbracket \quad &:\equiv \mathrm{pr}_1 s \quad \llbracket B, d, y \rrbracket \\
(\pi_2 s) \quad \llbracket B, d, y \rrbracket \quad &:\equiv \mathrm{pr}_2 s \quad \llbracket B, d, y \rrbracket \\
(s, t) \quad \llbracket B, d, y \rrbracket \quad &:\equiv \left(s \quad \llbracket B, d, y \rrbracket \quad, t \quad \llbracket B, d, y \rrbracket \quad\right) \\
(\mathrm{in}_1 s) \quad \llbracket B, d, y \rrbracket \quad &:\equiv \text{inl } s \quad \llbracket B, d, y \rrbracket \\
(\mathrm{in}_2 s) \quad \llbracket B, d, y \rrbracket \quad &:\equiv \text{inr } s \quad \llbracket B, d, y \rrbracket
\end{aligned}
$$

Now, with both polynomial type constructors and constructor terms in place, we can define HIT signatures.

Definition 1.8. ⚙ A *Van der Weide HIT signature of homotopy level* $S(Sh)$ consists of:

- a finite collection $(H_i)_{i:\ \mathrm{Fin}\ k}$ of k polynomials of universe level ℓ and homotopy level h (the point constructor argument types);
- a finite collection $\left(A_j, t_j, r_j\right)_{j:\ \mathrm{Fin}\ n}$ of n path constructors, all being such that:
  - A_j is a polynomial of universe level ℓ and homotopy level h (the argument type);
  - t_j and r_j (the endpoints) are path constructor terms from A_j to $\mathbb{X}$ over $(H_i)_{i:\ \mathrm{Fin}\ k}$.

In other words, a signature stores the information needed for an instance of the following HIT schema.

Let $T : h\text{–Type}$ be the HIT generated by:

- $c_1 : \mathrm{pr}_1 H_1[T] \to \mathrm{pr}_1 T$

  ...
- $c_k : \mathrm{pr}_1 H_k[T] \to \mathrm{pr}_1 T$
- $p_1 : \Pi_{x:\mathrm{pr}_1 A_1[T]} \quad t_1 \quad \llbracket T, \text{cluster } c, x \rrbracket \quad =_T r_1 \quad \llbracket T, \text{cluster } c, x \rrbracket$

  ...
- $p_n : \Pi_{x:\mathrm{pr}_1 A_n[T]} \quad t_n \quad \llbracket T, \text{cluster } c, x \rrbracket \quad =_T r_n \quad \llbracket T, \text{cluster } c, x \rrbracket$

Note that general constructors for higher paths are not available in this schema out-of-the-box.

1.1.2. W-Suspensions

While Van der Weide's higher inductive types manage to satisfy real-world common use cases in algebra and programming, they suffer from the theoretical shortcoming of not naturally arising as a generalisation of Martin-Löf's well-founded trees, commonly referred to as "$\mathcal{W}$-types" (P. Martin-Löf [18]). They also depend on ad-hoc grammars, which will make our proofs sometimes depend on ad-hoc inductions. Neither of these observations applies to our second working definition, $\mathcal{W}$ -suspensions by K. Sojakova [24], which subsume both $\mathcal{W}$-types and suspensions (The Univalent Foundations Program [27]) by design. This second working definition will be our starting point in the study of non-truncated higher inductive types. As before, we fix a universe level ℓ.

Definition 1.9. ⚙ A *$\mathcal{W}$-suspension signature* of universe level ℓ is a tuple $S :\equiv (A, B, C, l, r)$, with

- $A, C : \mathcal{U}_i$,
- $B : A \to \mathcal{U}_i$, and
- $l, r : C \to A$.

As our name choices suggest, this object is meant to extend a commonplace $\mathcal{W}$-type signature (A, B) with three new members. For starters, $C : \mathcal{U}_i$ indexes our path constructors, much like A indexes the point constructors. We can also talk about C being the type of "labels" for paths between points, much like A is the type of "labels" for points. Finally, $l : C \to A$ ($r : C \to A$) maps each path constructor to its left (right) endpoint. If we name the type induced by this signature W, then a path constructor takes as its two arguments terms $t : B(lc) \to W$ and $s : B(rc) \to W$. These are fed to the two point constructors labeled by lc and rc respectively. The two resulting terms of type W are the endpoints of the constructed path.

Unlike Van der Weide's HITs, $\mathcal{W}$-suspensions can therefore be infinitary, but only allow for a more restrictive form of path constructors.

1.1.3. Examples

A selection of examples of higher inductive types follows. The first one is can be seen as an instance of both our working definitions.

Example 1.10. ⚙ A textbook example of higher inductive type is the circle S^1. It is defined as the HIT generated by:

- $\mathrm{base} : S^1$;
- $\mathrm{loop} : \mathrm{base} =_{S^1} \mathrm{base}$.

Even when sticking to homotopy sets alone, higher inductive types can be very useful. They allow us to construct arbitrary free algebraic structures on a given homotopy set without coming up with an explicit construction.

Example 1.11. ⚙ Given a homotopy set $(A, k) : 0\text{–}\mathrm{Type}_\ell$, we can construct the free semigroup on it by simply "writing down" the semigroup operation and axiom. The free semigroup on A can be defined as the HIT generated by:

- $\eta : A \to \mathrm{FreeSemigroup}\ (A, k)$;
- $\star : \mathrm{FreeSemigroup}\ (A, k) \to \mathrm{FreeSemigroup}\ (A, k) \to \mathrm{FreeSemigroup}\ (A, k)$;
- $\mathrm{associative} : \prod_{a,b,c:\ \mathrm{FreeSemigroup}\ (A,k)} (a \star b) \star c =_{\mathrm{FreeSemigroup}\ (A,k)} a \star (b \star c)$;
- $\mathrm{truncated} : \mathrm{isSet}\ (\mathrm{FreeSemigroup}\ (A, k))$.

Note that the associative path constructor cannot be expressed directly in a $\mathcal{W}$-suspension signature. The same phenomenon will also occur multiple times in the next example.

Real world programming can also benefit from higher inductive types, as explored by H. Basold, H. Geuvers, and N. van der Weide [7]. In this context, HITs are used as "data types with laws".

Example 1.12. ⚙ Finite sets over a type A (à-la-Kuratowski) are defined by D. Frumin, H. Geuvers, L. Gondelman, and N. van der Weide [12] as the HIT $\mathcal{K}A$ generated by:

- $\emptyset : \mathcal{K}A$;
- $\{\cdot\} : A \to \mathcal{K}A$;

- $\cup : \mathcal{K}A \to \mathcal{K}A \to \mathcal{K}A$;
- $\mathrm{nl} : \prod_{x:\mathcal{K}A} \emptyset \cup x =_{\mathcal{K}A} x$;
- $\mathrm{nr} : \prod_{x:\mathcal{K}A} x \cup \emptyset =_{\mathcal{K}A} x$;
- $\mathrm{idem} : \prod_{a:A} \{a\} \cup \{a\} =_{\mathcal{K}A} \{a\}$;
- $\mathrm{assoc} : \prod_{x,y,z:\mathcal{K}A} x \cup (y \cup z) =_{\mathcal{K}A} (x \cup y) \cup z$;
- $\mathrm{com} : \prod_{x,y:\mathcal{K}A} x \cup y =_{\mathcal{K}A} y \cup x$;
- $\mathrm{trunc} : \mathrm{isSet}\ (\mathcal{K}A)$.

1.2. Impredicative Encodings

System F, the polymorphic lambda calculus, is expressive enough to allow for (finitary) inductive data types to be encoded in it, as illustrated by M. H. Sørensen and P. Urzyczyn [26]. The encodings are strictly guided by the elimination principle of the type we are trying to emulate. They are said to be "impredicative" because of the essential role played by the impredicative $\forall$ of System F, which quantifies over the type being encoded, too.

Example 1.13. The impredicative encoding for the type of natural numbers $\mathbb{N}$ in System F follows from its elimination principle.

$$\mathbb{N}_F :\equiv \forall X.(X \to X) \to X \to X$$

It comes with constructors and a recursor:

$$\begin{array}{lll} \mathbb{0}_F : \mathbb{N}_F & \mathbb{S}_F : \mathbb{N}_F \to \mathbb{N}_F & \mathrm{rec}_{\mathbb{N}_F} : \forall X.(X \to X) \to X \to \mathbb{N}_F \to \mathbb{N}_F \\ \mathbb{0}_F :\equiv \Lambda X.\lambda f.\mathrm{id}_X & \mathbb{S}_F :\equiv \lambda n.\Lambda X.\lambda f x.f(nXfx) & \mathrm{rec}_{\mathbb{N}_F} :\equiv \Lambda X.\lambda f x n.nXfx \end{array}$$

In a version of homotopy type theory with a bottom, impredicative universe, this kind of encoding does not fully work out: the η-equality is not satisfied, not even propositionally. As η-rules are unicity principles, their failure can be seen as a signal that the encoding features some non-standard terms (at least in some models). In this setting, this is equivalent to dependent elimination not being available. This was discussed by S. Awodey, N. Gambino, and K. Sojakova [5], [6]. "Large" elimination (i.e., elimination into types of higher universes) is not possible, either. S. Speight [25], and later S. Awodey, J. Frey, and S. Speight [4] propose a refinement to encode all set-level $\mathcal{W}$-types while retaining η-equality/dependent elimination into 0-types (thus, throwing the non-standard terms away). The actual construction for a generic $\mathcal{W}$-type is only spelled out by X. Ripoll Echeveste [21], S. Bronsveld [9] (both unpublished), and S. Bronsveld, H. Geuvers, and N. van der Weide [8]. X. Ripoll Echeveste [21], starting from ideas by M. Shulman [23], constructs a separate refinement, where dependent elimination spans the whole impredicative universe. None of these encodings allow for large elimination.

1.3. The System

Our base dependent type theory features dependent functions, strong sums, intensional identity, and function extensionality. The bottom universe $\mathcal{U}_0$ is assumed to be impredicative. So we have a predicative formation rule for successor universes, but an impredicative formation rule for the bottom universe.

$$\frac{\Gamma \vdash A : \mathcal{U}_{Si} \quad \Gamma, a : A \vdash B : \mathcal{U}_{Si}}{\Gamma \vdash \prod_{x:A} B : \mathcal{U}_{Si}} \qquad \frac{\Gamma, a : A \vdash B : \mathcal{U}_0}{\Gamma \vdash \prod_{x:A} B : \mathcal{U}_0}$$

The latter does not pose size restrictions on the index type when we are forming a product in the impredicative universe.

To be able to state Van der Weide's HIT schemas and rules only, we of course need to assume the relevant inductive types. These are of course not needed when working with $\mathcal{W}$-suspensions. Our untruncated encodings will also assume the existence of a natural numbers type. This is not assumed by our set-truncated encodings.

A few observations: although we invoke functional extensionality regularly, we never make use of the univalence axiom itself. So our results already hold within the intensional type theory by P. Martin-Löf [17], as long as we add function extensionality as an axiom. Still, homotopy levels, which we make use of in the next chapter alone, historically originate in homotopy type theory. On top of that, many of our examples are motivated by the "types-as-spaces" interpretation.

Precisely because we will not concern ourselves with univalence, from now on we use the word "category" to refer to HoTT book precategories. Additionally, because of impredicativity, in the following we can afford to work inside $\mathcal{U}_0$, rather than make our constructions parametric on a generic universe level ℓ like we have done so far. Because of this, we will even avoid explicitly mentioning universe level 0. For example, we will use $h\text{–}\mathrm{Type}$ in place of $h\text{–}\mathrm{Type}_0$, and Set in place of Set_0. Similarly, arguments declared in between braces when writing a subscript for a product type, and some arguments implicitly declared in English prose, will be considered to be "implicit". That is, they will not be explicitly declared in the corresponding lambda abstractions, nor they will be explicitly passed in the corresponding function applications. We only ever declare implicit arguments when their value is guaranteed to be inferable.

1.4. Contributions

This thesis consists of two main contributions:

1. a construction of Van der Weide's HITs defined in H. Basold, H. Geuvers, and N. van der Weide [7] as an impredicative encoding that eliminates into set-truncated types of the impredicative universe. This is fully formalised in Agda;
2. a construction of $\mathcal{W}$-suspensions as an impredicative encoding that eliminates into the impredicative universe.

The next chapter covers the former, while the chapter after that covers the latter. Finally, we dedicate one more chapter to a brief description of our formalisation.

Chapter 2

Set-Truncated Higher Inductive Types

In this chapter, we encode Van der Weide's higher inductive types as defined by H. Basold, H. Geuvers, and N. van der Weide [7] within the subuniverse of homotopy sets. That is, we start from HIT signatures that use 0-level polynomial type constructors, and we construct the corresponding 0-truncated higher inductive types. N. van der Weide and H. Geuvers [29] already developed initial semantics. Like them, we refer to *homotopy initiality*, i.e. hom-types being contractible, as simply "initiality", as this is the only notion of initiality we are interested in. We define the category of *lawful set algebras* for a given HIT signature in Section 2.1. In Section 2.2, we exploit impredicativity to conclude that this category has arbitrary limits. We use this fact in section Section 2.3 to construct the desired initial object in our category. Finally, Section 2.4 shows that initiality implies the desired β-rule and η-rule.

Example 2.1. ⚙ The exposition will be accompanied by a working example: natural numbers modulo 3 as a set-truncated higher inductive type $\mathbb{N}/3\mathbb{N}$, generated by the constructors:

- $0 : \mathbb{N}/3\mathbb{N}$;
- $S : \mathbb{N}/3\mathbb{N} \to \mathbb{N}/3\mathbb{N}$;
- $\mathrm{mod} : 0 =_{\mathbb{N}/3\mathbb{N}} S(S0)$.

A definition using normal inductive types would enumerate 0, 1, and 2 as three distinct constructors. As a consequence, the usual operations on such a type (e.g., addition, multiplication, …) would either require hard-coding or mappings to and from $\mathbb{N}$.

2.1. Lawful Set Algebras

We start from the usual notion of algebra over an endofunctor, which provides categorical semantics for $\mathcal{W}$-types.

Definition 2.2. ⚙ Given a category C, the type of *algebras over an endofunctor* $F : C \to C$ is

$$F\text{–Algebra} :\equiv \sum_{X :\, \mathrm{Ob}(C)} \mathrm{Hom}_C(FC, C).$$

We sometimes shorten "algebra over F" as "F-algebra". We refer to the first and second element of this dependent pair as *algebra carrier* and *algebra map* respectively.

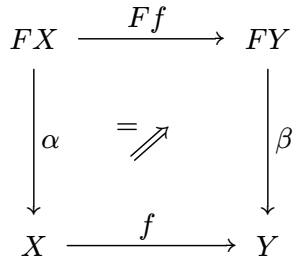
Example 2.3. `ℕ/3ℕ` has a 0-ary point constructor (or, equivalently, a point constructor with a single argument of the unit type) and a point constructor with a recursive argument. If we work in Set, the endofunctor bundling these point constructor arguments together is

$$F : \mathrm{Set} \to \mathrm{Set}$$
$$FX :\equiv 1 + X.$$

An F-algebra would consist of a set A equipped with a function $1 + A \to A$ designating a "zero" element in A as well as a "successor" function on A.

These algebras have their own notion of morphism, which ensures the structure picked by the starting algebra map is preserved.

Definition 2.4. ⚙ Given a category C and an endofunctor $F : C \to C$, an *F-algebra morphism* between F-algebras (X, α) and (Y, β) is a map $f : \mathrm{Hom}_C(X, Y)$ equipped with a witness for the commutative diagram



.

Because hom-sets are homotopy sets in our working definition of category, the type of the witness mentioned in this definition is a mere proposition. So, when checking two algebra morphisms for equality, it is always enough to check for equality of the two underlying functions.

Example 2.5. In our previous example, algebra morphisms would be maps between carriers that preserve the "zero" element and commute with the "successor" function.

As it is to be expected, algebras over an endofunctor form a category.

Proposition 2.6. ⚙ If $F : C \to C$ is an endofunctor over a category C, then F-algebras and F-algebra morphisms among them form a category, that we name F-Alg.

Since we are dealing with 0-truncated types, we focus on endofunctors on the Set category[5], and thus talk of *set algebras* and F– SetAlg. Because algebra morphisms preserve the structure of algebras, they commute with the evaluation of a path constructor term.

Proposition 2.7. ⚙ Let $(H_i)_{i:\ \mathrm{Fin}\ k}$ be a family of polynomials for homotopy sets. We set $F :\equiv \bigoplus H$. Let $(f, h) : \mathrm{Hom}_{F\text{–SetAlg}}((A, \alpha), (B, \beta))$. Give polynomials for homotopy sets G and P, for any $r :$ PathConstructorTerm HGP and $x : \mathrm{pr}_1 G[A]$, we have

$$P[[f]]r \quad 【A, \alpha, x】 \quad =_{\mathrm{pr}_1 P[B]} \ r \quad 【B, \beta, G[[f]]x】 \quad .$$

Proof. By induction on r:

- if r is a constant or variable 𝓍, the statement holds definitionally;
- if r is an applied binary projection or sum constructor, we can apply the relevant action on paths to our induction hypothesis;

[5]This is the category of terms of type 0– Type and functions between their first components.

- if r is a fully applied pair constructor, we check for component-wise equality, which holds due to the induction hypothesis;
- if $r \equiv c_i s$ for some $i : \mathrm{Fin}\ k$ and $s : \mathrm{PathConstructorTerm}\ HGH_h$, we have

$$\begin{aligned} \mathrm{scatter}\ \alpha i s\ \ \llbracket\ A, \alpha, x\ \rrbracket\ \ &=_B \beta\big(F[[f]]\big(\mathrm{in}_i s\ \ \llbracket A, \alpha, x \rrbracket\ \ \big)\big) && \langle h \rangle \\ &\equiv \beta\big(\mathrm{in}_i\big(H_i[[f]] s\ \ \llbracket A, \alpha, x \rrbracket\ \ \big)\big) \\ &\overset{\mathrm{IH}}{=}_B \mathrm{scatter}\ \beta i s\ \ \llbracket B, \beta, G[[f]] x \rrbracket\ \ . \end{aligned}$$

□

To provide semantic counterparts to path constructors, we equip set algebras with appropriate families of paths, thus getting to the notion of *lawful algebras over a HIT signature.*

Definition 2.8. ⚙ Let $S \coloneqq \Big(\big(k, (H_i)_{i:\ \mathrm{Fin}\ k}\big), \Big(n, \big(A_j, t_j, r_j\big)_{j:\ \mathrm{Fin}\ n}\Big)\Big)$ be a HIT signature for homotopy sets. The type of *lawful set algebras over HIT signature* S is

$$S\text{–}\,\mathrm{LawfulAlgebra} \coloneqq \sum_{\mathcal{A}:(\mathrm{polyFuncOnSet}\ (\bigoplus H))\text{–}\mathrm{Algebra}} \mathrm{isLawful}\ \mathcal{A}$$

where

$$\mathrm{isLawful}\ (X, \alpha) \coloneqq \prod_{j:\ \mathrm{Fin}\ n}\ \prod_{x:\ \mathrm{fst}\ (A_j[X])} t_j\ \ \llbracket X, \alpha, x \rrbracket\ \ =_{\mathrm{fst}\ X} r_j\ \ \llbracket X, \alpha, x \rrbracket\ \ .$$

The lawfulness condition is a mere proposition, so lawful set algebras form the full subcategory LawSetAlg of set algebras.

Proposition 2.9. ⚙ The lawfulness condition for set algebras is a mere proposition.

Proof. Obvious, as the carrier of a set algebra is a homotopy set. □

Corollary 2.10. ⚙ Let S be a HIT signature for homotopy sets, and F the polynomial induced by the $\oplus$-fold over the path constructor arguments of S. Then, the lawful set algebras induced by S form a full subcategory of the set algebras over F.

Not requiring any additional conditions from our morphisms is only natural: since we are working with homotopy sets, all paths are trivially preserved. This is also reflected by the "set-truncated" η -rule we state and prove at the end of the chapter. In the next section, we investigate limits within the category of lawful set algebras.

2.2. Limits in the Lawful Set Algebras Category

In an impredicative setting, Set, the category of small homotopy sets and functions between them, is not just complete: it has limits of arbitrary shapes, which need not be small, unlike the objects of this category. This is because the usual construction of a limit by products and equalisers is not limited by the products' predicativity any more.

Definition 2.11. ⚙ Given a category $\mathbb{J}$ of any size and a functor $D : \mathbb{J} \to \mathrm{Set}$, we define the *limit encoding for* D as

$$\lim_{\leftarrow i} D_i : 0\text{– Type}$$

$$\lim_{\leftarrow i} D_i :\equiv \left(\sum_{\phi : D^*} \text{Nat } \phi, h\right)$$

where

$$D^* :\equiv \prod_{i:\mathbb{J}_0} Di$$

and Nat is the naturality condition

$$\text{Nat } \phi :\equiv \prod_{\{i,j:\mathbb{J}_0\}} \prod_{u:\text{Hom}_{\mathbb{J}}(i,j)} Du(\phi i) =_{Dj} \phi j.$$

As a proof h witnessing the set-truncation for the first projection, we observe that D^* is a set (it is a product of sets) and, fixed $\phi : D^*$, Nat ϕ is a proposition (it is a product of products of propositions).

To give our limit encodings the structure of cones, we equip them with the obvious projections, which are just the ones from the underlying product.

Definition 2.12. ⚙ Let $D : \mathbb{J} \to \text{Set}$ be a functor. We define μ_D, the *limit encoding cone on* $\lim_{\leftarrow i} D_i$, as the following D-cone with vertex $\lim_{\leftarrow i} D_i$.

$$\mu_D : \text{Cone} \quad D \quad \lim_{\leftarrow i} D_i$$

$$\mu_D :\equiv (\pi_j \circ \text{pr}_1 \circ \text{pr}_1, k)$$

where k, our proof of the cone commutativity condition, is

$$\lambda i, j : \mathbb{J}_0, u : \text{Hom}_{\mathbb{J}}(i,j).\ \text{funExt } \lambda(\phi, \vartheta) : \lim_{\leftarrow i} D_i.\vartheta i j u.$$

We now proceed to prove that our cone is, in fact, limiting.

Theorem 2.13. ⚙ The cone μ_D described above for a generic category J of arbitrary size and any functor $D : J \to \text{Set}_{\mathcal{U}_0}$ is always terminal.

Proof. Given a D-cone with a set V (equipped with a proof of is– 0– Type V) as vertex and a family of projections ν, we need to define a map $f : V \to \text{pr}_1 \lim_{\leftarrow i} D_i$. We also need to prove that it is a cone morphism, and that it is the only cone morphism with such source and target. Such a universal map is constructed using the projections of the source cone.

$$f : V \to \text{pr}_1 \lim_{\leftarrow i} D_i$$

$$fv :\equiv \big((\lambda j : \mathbb{J}_0.\nu_j v), h\big)$$

As a proof h for naturality, let $u : \text{Hom}_{\mathbb{J}}(i,j)$. We are to show $u(\nu_i v) =_{\text{pr}_1(Dj)} \nu_j v$. By definition of cone, we already have $u \circ \nu_i =_{V \to Dj} \nu_j$. We turn this to a homotopy and apply it to v. Of course, f is a cone morphism, as the relevant equalities all hold definitionally. In Set, "being a cone morphism" is a mere proposition, so when proving unicity of our construction, it is enough to check for equality between f and the underlying function $g : V \to \text{pr}_1 \lim_{\leftarrow i} D_i$ of some other parallel cone morphism. We

prove this by function extensionality, fixing $v : V$. As already observed in Definition 2.11, naturality in Set is always a proposition, so we can use function extensionality again fixing $j : \mathbb{J}_0$. Now we only need to check $\nu_j v =_{\mathrm{pr}_1 Dj} \mathrm{pr}_1(gv)j$. Because g is a cone morphism, we can just take the relevant commutativity condition for j, turn it into a homotopy and apply it to v. □

Corollary 2.14. ⚙ Set has limits of any shape, regardless of size, constructed as above.

Proof. This follows immediately from our previous proof of terminality and the definition of limit. □

It is a known fact that the category of set algebras over an endofunctor inherits all limits from Set. This is shown by observing that it is possible to compute each limit pointwise. To prove the same result for the category of lawful set algebras induced by a HIT signature, we'll need an appropriate forgetful functor.

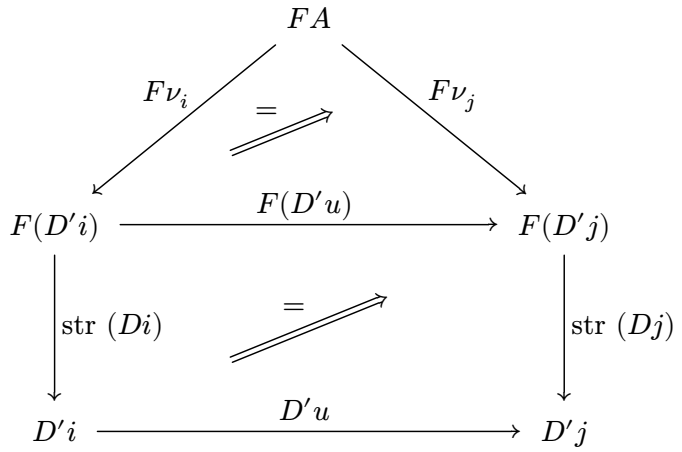
Definition 2.15. ⚙ Let S be a HIT signature for homotopy sets, and F the polynomial induced by the $\oplus$-fold over the path constructor arguments of S. We define ForgetLawSetAlg as the forgetful functor from S– LawSetAlg to Set obtained by composing the forgetful functor $\mathbb{U} : F$– SetAlg $\to$ Set after the inclusion functor $\iota : S$– LawSetAlg $\to F$– SetAlg.

We are now ready to prove the main theorem of this section. The proof strategy, similar to what is done for regular algebras, is to use our forgetful functor so that we can build our limit starting from the corresponding one in Set.

Theorem 2.16. ⚙ Let S be a HIT signature for homotopy sets. Then S– LawSetAlg has limits of any shape.

Proof. We work with lawful algebras on HIT signature $S :\equiv \left((H_i)_{i:\ \mathrm{Fin}\ k}, (A_j, t_j, r_j)_{j:\ \mathrm{Fin}\ n}\right)$. Much like before, fixed a category J of arbitrary size and a functor $D : J \to \mathrm{Set}$, we shall construct a lawful set algebra over S as a vertex of a D-cone. After doing that, we will be left with proving terminality of such a cone. Note that composing ForgetLawSetAlg after D yields a functor $D' : J \to$ Set. Consider $V :\equiv \varprojlim_i D'_i$, the limit encoding for D', and the corresponding limit encoding cone M. By Theorem 2.13, we have a proof for its terminality as a D'-cone. We denote the family of projections from V by μ, and use H_μ to refer to the evidence for the cone condition on M. Finally, we use $F : \mathrm{Set} \to \mathrm{Set}$ for the endofunctor induced by $\bigoplus H$.

Now, observe that any D'-cone with vertex $A : 0$– Type and projections ν can be turned into a D'-cone with vertex FA, as shown in the following diagram. In it, the top triangle commutes since (A, ν) is a cone, and the bottom square commutes because $D'u$ is an F-algebra morphism.

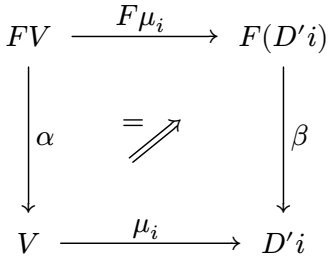


We make use of this trick immediately to construct a vertex for the D-cone. We select V as the carrier. As a map for the F-algebra, we choose the function $\alpha : \mathrm{pr}_1 FV \to \mathrm{pr}_1 V$ underlying the universal cone morphism from the D'-cone constructed as above (set A to V) to the terminal cone M. To check (V, α)'s lawfulness for fixed values of $j : \mathrm{Fin}\ k$ and $x : A_j[V]$, we use function extensionality, since we already stated that naturality is a mere proposition in this context. Then, let $i : \mathbb{J}_0$. If Di is a lawful algebra with carrier B and map $\beta : FB \to B$, the desired equality holds.

$$
\begin{aligned}
\mathrm{pr}_1 t_j \quad 【\ V, \alpha, x\ 】 \quad i &=_B t_j \quad 【\ B, \beta, A_j[[(\lambda q.\mathrm{pr}_1 qi)]]x\ 】 && \langle \text{distributivity} \rangle \\
&=_B r_j \quad 【\ B, \beta, A_j[[(\lambda q.\mathrm{pr}_1 qi)]]x\ 】 && \langle Di\text{'s lawfulness} \rangle \\
&=_B \mathrm{pr}_1 r_j \quad 【\ V, \alpha, x\ 】 \quad i && \langle \text{distributivity} \rangle
\end{aligned}
$$

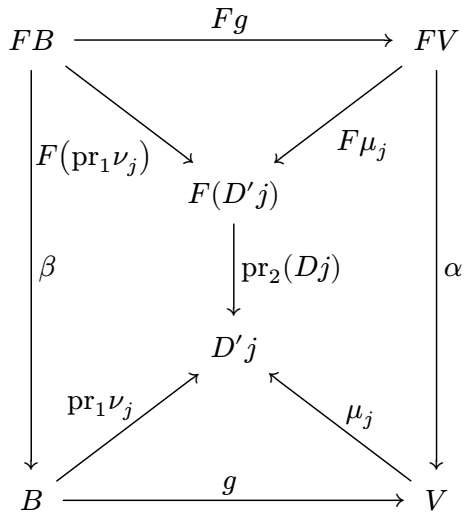
The distributivity property mentioned in this equality chain is just an instance of Proposition 2.7. Indeed, $\lambda q : \mathrm{pr}_1 V.\mathrm{pr}_1 qi$ is an F-algebra morphism from, due to α being a cone morphism.

So our vertex is actually $((V, \alpha), h)$, where h is the lawfulness proof just provided. As projection to $D'i$, with $i : \mathbb{J}_0$, we reuse μ_i. Again, this is surely an algebra morphism, because α being a cone morphism ensures that the following diagram commutes, where β is the map of lawful algebra Di.



Equipped with μ, $((V, \alpha), h)$ is indeed a cone: because "being an algebra morphism" is a mere proposition as observed above, H_μ is already evidence of this. So we are left with proving terminality. Let $\mathbb{B}$ be a lawful algebra with carrier $B : 0\text{–}\mathrm{Type}$ and algebra map $\beta : FB \to B$. We fix a D-cone with $\mathbb{B}$ as its vertex given by a family ν of projections. The desired universal arrow is a cone morphism from this generic cone to the one we defined earlier. We can turn the D-cone with vertex $\mathbb{B}$ into a D' -cone with vertex B using ForgetLawSetAlg. We denote the family of projections for the latter ν'.

There's a unique D'-cone morphism to the terminal D'-cone with vertex V and projections family μ. We denote the underlying function by $g : B \to V$. Again, algebras morphisms form a subtype of functions, so if we can manage to reuse g as a D-cone morphism, too, we would get the uniqueness and cone morphism commutativity conditions for free. All that is really left to do is proving that g actually behaves like an algebra morphism. We do so by using the following diagram family, which is indexed by $j : \mathbb{J}_0$.



In the diagram:

- the bottom triangle commutes due to g being a D'-cone morphism;
- the top triangle commutes because of F's action on the bottom triangle;
- the left trapezoid commutes as witnessed by $\mathrm{pr}_2\nu_j$;
- the right trapezoid commutes since μ_j is an algebra morphism, as illustrated by the second diagram.

Our goal is making the external rectangle commute though. To do so, it is enough to observe that both $g \circ \beta$ and $\alpha \circ Fg$ can be shown to be cone morphisms to the terminal D'-cone. The source cone, in this case, has vertex FB and is constructed from cone (B, ν') using the trick illustrated by the first diagram of this proof. The cone morphism commutativity condition for both $g \circ \beta$ and $\alpha \circ Fg$ is easily checked via diagram chasing on the last diagram. This concludes the proof of terminality. □

Next, we shall make use of one particular limit, the one of the identity functor, to construct the initial lawful set algebra.

2.3. Initial Lawful Set Algebra

To construct the initial lawful set algebra, we make use of the following well-known characterisation result for the initial object of a generic category.

Theorem 2.17. ⚙⚙ Let I be an object in some category $\mathcal{C}$. Then, I is initial if and only if it is the limit of $\mathrm{id}_{\mathcal{C}}$, the identity functor on $\mathcal{C}$.

Proof. From left to right: given an object X, we can take the universal map from $p_X : I \to X$ as the corresponding projection. Given a second object Y and a morphism $f : \mathrm{Hom}_{\mathcal{C}}(X, Y)$, we are assured

$f \circ p_X =_{\mathrm{Hom}_{\mathcal{C}}(I,Y)} p_Y$ by the uniqueness of universal map p_Y. Then, (I, p) behaves as a cone. Given a second $\mathrm{id}_{\mathcal{C}}$-cone (J, q), we consider $q_I : \mathrm{Hom}_{\mathcal{C}}(J, I)$. This is a cone morphism between (J, q) and (I, p): fixed an object X, $p_X \circ q_I =_{\mathrm{Hom}_{\mathcal{C}}(J,X)} q_X$ holds since (J, q) is a cone. Such a cone morphism is necessarily unique: if there were a second cone morphism $y : J \to I$, it would satisfy

$$y =_{\mathrm{Hom}_{\mathcal{C}}(J,I)} 1_I \circ y =_{\mathrm{Hom}_{\mathcal{C}}(J,I)} p_I \circ y =_{\mathrm{Hom}_{\mathcal{C}}(J,I)} q_I.$$

From right to left: let $x : \mathrm{Ob}(\mathcal{C})$. If I is the limit of the identity functor, it comes equipped with a projection $p_X : I \to X$. To show uniqueness of such a map, observe that $p_I : I \to I$ is a cone endomorphism. That's because $p_Y \circ p_I =_{\mathrm{Hom}_{\mathcal{C}(I,Y)}} p_Y$ holds for a generic $Y : \mathrm{Ob}(\mathcal{C})$, as (I, p) is a cone. Furthermore, (I, p) is terminal, so p_I must coincide with the endomorphism underlying the identity cone morphism on (I, p), and that's just 1_I. Now then, let $y : I \to X$. We have

$$y =_{\mathrm{Hom}_{\mathcal{C}}(I,X)} y \circ 1_I =_{\mathrm{Hom}_{\mathcal{C}}(I,X)} y \circ p_I =_{\mathrm{Hom}_{\mathcal{C}}(I,X)} p_X$$

as desired. □

This means that the work carried on in the previous section was enough to obtain the initial lawful set algebra.

Corollary 2.18. ⚙ Let S be a HIT signature for homotopy sets. Then S–LawSetAlg has an initial object.

Proof. Follows immediately from Theorem 2.16 and Theorem 2.17. □

In the next section, we show that the initial lawful set algebra indeed satisfies the desired rules.

2.4. Rules

In this section, we work with a generic HIT signature $S \coloneqq \left(\left(k, (H_i)_{i:\ \mathrm{Fin}\ k}\right), \left(n, (A_j, t_j, r_j)_{j:\ \mathrm{Fin}\ n}\right)\right)$. As before, we use F for the endofunctor on Set induced by $\bigoplus H$. To even be able to state the rules whose validity we wish to prove, we must first explain how to recover the encoding of our higher inductive type, as well as those for the corresponding constructors. As it is usual for initial algebra semantics, the encoding of the type is provided by the carrier.

Definition 2.19. ⚙ The *encoding of the set-truncated higher inductive type given by* S, denoted by $T_S : 0\text{–}\mathrm{Type}$, is defined as the carrier of the initial lawful set algebra over S as constructed in Corollary 2.18. Written explicitly, $T_S :\equiv \left(\sum_{\phi : \prod_{((A,\alpha),h):S\text{–LawSetAlg}} A} \mathrm{Nat}\ \phi, k\right)$

where k is the proof of the first member being a homotopy set as constructed in Definition 2.11.

Point constructors also follow the usual pattern, i.e. they are trivially derived from the algebra map.

Definition 2.20. ⚙ The *encodings of the point constructors given by* S, or c_S, are defined by composing scatter after the algebra map of the initial lawful set algebra as defined in Corollary 2.18. Explicitly, this can be written as

$$c_S : \prod_{i:\,\mathrm{Fin}\,k} \mathrm{pr}_1 H_i[T_S] \to \mathrm{pr}_1 T_S$$

$$\begin{aligned} c_S :\equiv \mathrm{scatter}\ \lambda x : \mathrm{pr}_1(FT_S).\Big(& \\ & \lambda((A,\alpha),h) : S\text{–}\mathrm{LawSetAlg}_0.\alpha(F(\lambda y : \mathrm{pr}_1 T_S.\mathrm{pr}_1 y((A,\alpha),h))x), \\ & \lambda(f,p) : \mathrm{Hom}_{S\text{–}\mathrm{LawSetAlg}_0}(((A,\alpha),h),((B,\beta),k)).\ \mathrm{happly}\ (...f)x \\ \Big). & \end{aligned}$$

The ellipsis ("...") replaces the proof for the commutativity condition of the cone

$$(FT_S, \lambda((A,\alpha),h).\alpha \circ F(\lambda x : \mathrm{pr}_1 T_S.\mathrm{pr}_1 x((A,\alpha),h))$$

as described in Theorem 2.16.

The real novelty are the encodings for the path constructors. These are of course implemented using the lawfulness proof.

Definition 2.21. ⚙ The *encodings of the path constructors given by* S, or

$$p_S : \prod_{j:\,\mathrm{Fin}\,n} \prod_{x:\mathrm{pr}_1 A_j[T_S]} t_j \quad 【\ T_S, \mathrm{cluster}\ c_S, x】 \quad =_{\mathrm{pr}_1 T_S} r_j \quad 【\ T_S, \mathrm{cluster}\ c_S, x\ 】\ ,$$

are defined as the transport of the lawfulness proof for the initial lawful set algebra from Corollary 2.18 along the proof of cluster being a retract of scatter.

Explicitly writing down this term is not really important, as lawfulness at the set level is a mere proposition.

Finally, a *recursor* is defined.

Definition 2.22. ⚙ The *encoding of the recursor given by* S is

$$\begin{aligned} \mathrm{rec}_S : & \prod_{X:0\text{–}\mathrm{Type}} \prod_{d:\prod_{i:\,\mathrm{Fin}\,k} H_i[X]\to \mathrm{pr}_1 X} \left(\prod_{j:\,\mathrm{Fin}\,n} \prod_{x:\mathrm{pr}_1 A_j[X]} t_j \quad 【\ X, \mathrm{cluster}\ d, x】 \quad =_{\mathrm{pr}_1 X} r_j \quad 【\ X, \mathrm{cluster}\ d, x\ 】 \right) \\ & \to T_S \to \mathrm{pr}_1 X \\ \mathrm{rec}_S :\equiv & \lambda X d q t.\mathrm{pr}_1 t((X, \mathrm{cluster}\ d), q). \end{aligned}$$

As expected, the recursor is nothing but a curried version of the family of projections from the vertex of our limiting cone. So, even if not under this name, it has already appeared over and over in our exposition. Now, all the elements needed to state the β-rules are in place.

Proposition 2.23. ⚙ The following *point constructor* β*-rule for the higher inductive type induced by signature* S holds definitionally, for any $X : 0\text{–}\mathrm{Type}$, $x : \mathrm{pr}_1 H_i[T_S]$, $\left(d : \prod_{i:\,\mathrm{Fin}\,k} \mathrm{pr}_1 H_i[X] \to \mathrm{pr}_1 X\right)_{i:\,\mathrm{Fin}\,k}$, $\left(q : \prod_{j:\,\mathrm{Fin}\,n} \prod_{x:\mathrm{pr}_1 A_j[X]} t_j \quad 【\ X, \mathrm{cluster}\ d, x\ 】 \quad =_{\mathrm{pr}_1 X} r_j \quad 【X, \mathrm{cluster}\ d, x】\ \right)_{j:\,\mathrm{Fin}\,n}$.

$$\mathrm{rec}_S X d q\big(c_{S,i} x\big) \equiv d_i(H_i[[\mathrm{rec}_S X d q]]x)$$

Proof.

$$\begin{aligned}
\mathrm{rec}_S X dq(c_{S,i}x) &\equiv \mathrm{pr}_1(c_{S,i}x)((X, \text{cluster } d), q) && \langle \text{unfold } \mathrm{rec}_S \rangle \\
&\equiv \text{cluster } d(F(\lambda y : T_S.\mathrm{pr}_1 y((X, \text{cluster } d), q))(\mathrm{in}_i x)) && \langle \text{unfold } c_{S,i} \rangle \\
&\equiv \text{cluster } d(\mathrm{in}_i(H_i[[\lambda y : T_S.\mathrm{pr}_1 y((X, \text{cluster } d), q)]]x)) && \langle k\text{-ary sums' } \beta\text{-rule} \rangle \\
&\equiv \text{cluster } d(\mathrm{in}_i(H_i[[\mathrm{rec}_S X dq]]x)) && \langle \text{fold } \mathrm{rec}_S \rangle \\
&\equiv d_i(H_i[[\mathrm{rec}_S X dq]]x). && \langle \text{fold cluster} \rangle
\end{aligned}$$

□

Because T_S is a homotopy set, the corresponding rule for path constructors automatically holds propositionally, which is usually the desired kind of equality in this case, as argued by the The Univalent Foundations Program [27].

Remark. The following *path constructor β-rule for the higher inductive type induced by signature S* holds propositionally, for any $X : 0\text{–Type}$, $x : \mathrm{pr}_1 A_j[(T_S, h)]$ (h being proof of T_S being a homotopy set as constructed in Definition 2.11), $\left(d : \prod_{i:\ \mathrm{Fin}\ k} \mathrm{pr}_1 H_i[X] \to \mathrm{pr}_1 X\right)_{i:\ \mathrm{Fin}\ k}$, $\left(q : \prod_{j:\ \mathrm{Fin}\ n} \prod_{x:\mathrm{pr}_1 A_j[X]} t_j \quad \llbracket\, X, \text{cluster } d, x \,\rrbracket \quad =_{\mathrm{pr}_1 X} r_j \quad \llbracket X, \text{cluster } d, x \rrbracket \right)_{j:\ \mathrm{Fin}\ n}$.

$$\mathrm{ap}_{\mathrm{rec}_S X dq}(p_{S,j}x) =_{\mathrm{pr}_1 X} q_j(A_j[[\mathrm{rec}_S X dq]]x)$$

As for the η-rule, we use its instance corresponding to the identity function on our higher inductive type as a lemma which we will later invoke to prove the full-fledged rule.

Lemma 2.24. ⚙ The following *weak η-rule for the higher inductive type induced by signature S* holds propositionally.

$$\mathrm{rec}_S T_S c_S p =_{\mathrm{pr}_1 T_S \to \mathrm{pr}_1 T_S} \mathrm{id}_{\mathrm{pr}_1 T_S}$$

Proof. By function extensionality. When building the desired homotopy between the two sides of the equality, we destruct the input as (ϕ, φ) via Σ-induction. However, since naturality is a proposition at the set level, we just need to check

$$\mathrm{pr}_1(\mathrm{rec}_S T_S c_S p(\phi, \varphi)) =_{\prod_{((A,\alpha),h):S\text{–LawSetAlg}} A} \phi$$

for any φ. We now use function extensionality (and Σ-induction) again, fixing $((B, \beta), h) : S\text{–LawSetAlg}$ and proving

$$\mathrm{pr}_1(\mathrm{rec}_S T_S c_S p(\phi, \varphi))((B, \beta), h) =_{\mathrm{pr}_1 B} \phi((B, \beta), h)$$

Recall from the construction of the projections in Theorem 2.16 that $\lambda x : T_S.\mathrm{pr}_1 x((B, \text{cluster } (\text{scatter } \beta)), h') \equiv \mathrm{rec}_S B(\text{scatter } \beta)h'$ (where h' is h transported appropriately) is an F-algebra morphism. So by φ we have:

$$
\begin{aligned}
\mathrm{pr}_1(\mathrm{rec}_S T_S c_S p_S(\phi,\varphi))((B,\beta),h) &=_{\mathrm{pr}_1 B} \mathrm{pr}_1(\mathrm{rec}_S T_S c_S p_S(\phi,\varphi))((B,\mathrm{cluster}\ (\mathrm{scatter}\ \beta)),h') \\
&\equiv \mathrm{pr}_1(\phi((T_S,\mathrm{cluster}\ c_S),p_S))((B,\mathrm{cluster}\ (\mathrm{scatter}\ \beta)),h') \\
&=_{\mathrm{pr}_1 B} \mathrm{pr}_1(\phi((T_S,\mathrm{cluster}\ c_S),p_S))((B,\mathrm{cluster}\ (\mathrm{scatter}\ \beta)),h') \\
&=_{\mathrm{pr}_1 B} \mathrm{pr}_1(\phi I)((B,\mathrm{cluster}\ (\mathrm{scatter}\ \beta)),h') \\
&=_{\mathrm{pr}_1 B} \mathrm{pr}_1(\phi I)(B,\beta),h) \\
&=_{\mathrm{pr}_1 B} \phi((B,\beta),h) && \langle\varphi\rangle
\end{aligned}
$$

where I is the initial lawful set algebra. □

Generalising to an arbitrary (non-dependent) function from id_{T_S} is now very easy. Note how the premise can be read as stating that f behaves like an algebra morphism from our initial lawful set algebra to $(X,\mathrm{cluster}\ d)$. Indeed, this premise shall motivate our use of naturality.

Proposition 2.25. ⚙ The following *η-rule for the higher inductive type induced by signature* S holds propositionally, for any $X : 0\text{–}\mathrm{Type}$, $f : \mathrm{pr}_1 T_S \to \mathrm{pr}_1 X$, $\left(d : \prod_{i:\ \mathrm{Fin}\ k} \mathrm{pr}_1 H_i[X] \to \mathrm{pr}_1 X\right)_{i:\ \mathrm{Fin}\ k}$, $\left(q : \prod_{j:\ \mathrm{Fin}\ n} \prod_{x:\mathrm{pr}_1 A_j[X]} t_j \quad 【\ X,\mathrm{cluster}\ d,x\ 】 \quad =_{\mathrm{pr}_1 X} r_j \quad 【X,\mathrm{cluster}\ d,x】 \quad \right)_{j:\ \mathrm{Fin}\ n}$.

$$\left(f\circ \mathrm{cluster}\ c_S =_{\mathrm{pr}_1 F[T_S]\to \mathrm{pr}_1 X} \mathrm{cluster}\ d\circ Ff\right) \to \mathrm{rec}_S X d q =_{\mathrm{pr}_1 T_S \to \mathrm{pr}_1 X} f.$$

Proof. By function extensionality. Let $x : \mathrm{pr}_1 T_S$. We have:

$$
\begin{aligned}
\mathrm{rec}_S X d q x &\equiv \mathrm{pr}_1 x((X,\mathrm{cluster}\ d),q) && \langle\text{unfold } \mathrm{rec}_S\rangle \\
&=_{\mathrm{pr}_1 X} f(\mathrm{pr}_1 x((T_S,\mathrm{cluster}\ c_S),p_S)) && \langle \mathrm{pr}_1 x \text{ is natural}\rangle \\
&\equiv f(\mathrm{rec}_S T_S c_S p_S x) && \langle\text{fold } \mathrm{rec}_S\rangle \\
&=_{\mathrm{pr}_1 X} fx. && \langle\text{Lemma 2.24}\rangle
\end{aligned}
$$

□

As shown by S. Awodey, N. Gambino, and K. Sojakova [5], [6], this result is enough to derive the desired dependent eliminator.

Chapter 3

General Higher Inductive Types

In this chapter, inspired by the work of X. Ripoll Echeveste [21], we assume the existence of a natural numbers type $\mathbb{N}$ within our type system with the aim of dropping the homotopy level restrictions. This time around, we work with $\mathcal{W}$-suspensions by K. Sojakova [24]. Much like we have been doing insofar, we leave the word "homotopy" implicit when talking about homotopy initiality: this is because we will not need any other notion of initiality.

In Section 3.1, we develop the path algebra constructions needed for our work. Then, in Section 3.2, we recall the initial semantics for $\mathcal{W}$-suspensions. Section 3.3 shows how the naïve encoding can be refined to recover the same naturality property described in the previous chapter. Because we are not working with homotopy sets alone, naturality is not enough to obtain initiality any more. Hence, we reiterate our refinement in Section 3.4 and conclude showing initiality in Section 3.5.

3.1 Path Algebra Tools

In this chapter, all path algebra notation which is not explicitly introduced is borrowed from The Univalent Foundations Program [27]. Our path composition operator ($\blacksquare$) will be right-associative, but we will sometimes still use unnecessary parenthesis to help the reader. The functoriality of the action of a map on a path is left as an exercise for the reader in the HoTT book. We "solve" such exercise by providing explicit proof terms for it, so that we can use the induced definitional computation rules (without even stating the most obvious).

Lemma 3.1. (Lemma 2.2.2 from The Univalent Foundations Program [27].) For functions $f : A \to B$ and $g : B \to C$ and paths $p : x =_A y$ and $q : y =_A z$, we have:

1. ap-trans $fpq : \mathrm{ap}_f(p \blacksquare q) = \mathrm{ap}_f p \blacksquare \mathrm{ap}_f q$;
2. ap-inv $fp : \mathrm{ap}_f p^{-1} = \left(\mathrm{ap}_f p\right)^{-1}$;
3. ap-comp $gfp : \mathrm{ap}_g\left(\mathrm{ap}_f p\right) = \mathrm{ap}_{g \circ f} p$;
4. ap-id $p : \mathrm{ap}_{\mathrm{id}_A} p = p$.

Proof.

1. By based path induction on p and q. We set ap-trans $f\mathrm{refl}_y\mathrm{refl}_y :\equiv \mathrm{refl}_{\mathrm{refl}_{fy}}$;
2. by path induction on p. We set ap-inv $f\mathrm{refl}_x :\equiv \mathrm{refl}_{\mathrm{refl}_{fx}}$;
3. by path induction on p. We set ap-comp $f\mathrm{refl}_x :\equiv \mathrm{refl}_{\mathrm{refl}_{g(fx)}}$;
4. by path induction on p. We set ap-id $\mathrm{refl}_x :\equiv \mathrm{refl}_{\mathrm{refl}_x}$.

□

Similarly, we will also regularly invoke the following transport lemmas. In order to use these results within proof terms, we choose names for them.

Lemma 3.2. (Lemma 2.3.9 from The Univalent Foundations Program [27].) Given $P : A \to \mathcal{U}_0$ with $p : x =_A y$ and $q : y =_A z$ while $u : Px$, we have

$$\text{tr-trans } qpu : q_*(p_*u) =_{Pz} (p \centerdot q)_*u.$$

Lemma 3.3. (Lemma 2.3.10 from The Univalent Foundations Program [27].) For a function $f : A \to B$ and a type family $P : B \to \mathcal{U}_0$, and any $p : x =_A y$ and $u : P(fx)$, we have

$$\text{tr-comp } fpu : \text{transport}^{P\circ f} pu =_{P(fy)} \text{transport}^{P}\big(\text{ap}_f p\big)u.$$

Lemma 3.4. (Lemma 2.3.11 from The Univalent Foundations Program [27].) For $P, Q : A \to \mathcal{U}_0$ and a family of functions $f : \prod_{x:A} Px \to Qx$, and any $p : x =_A y$ and $u : Px$, we have

$$\text{tr-fun } fpu : \text{transport}^{Q} p(f_x u) =_{Qy} f_y(\text{transport}^{P} pu).$$

The next properties, on the other hand, are new. We define them for convenience.

Lemma 3.5. For parallel functions $f, g : A \to B$, a homotopy $h : f \sim g$ between them, function $e : Z \to A$, and consecutive paths $p : u =_A v$, $q : v =_A w$, $r : w =_A x$, $s : x =_A y$, $t : y =_A z$, we have:

1. surr-refl $p : \text{refl}_u \centerdot p \centerdot \text{refl}_v = p$;
2. ap-trans3 $fpqr : \text{ap}_f(p \centerdot q \centerdot r) = \text{ap}_f p \centerdot \text{ap}_f q \centerdot \text{ap}_f r$;
3. assoc5 $pqrst : p \centerdot (q \centerdot r \centerdot s) \centerdot t = (p \centerdot q) \centerdot r \centerdot (s \centerdot t)$;
4. inv-trans $pq : (p \centerdot q)^{-1} = q^{-1} \centerdot p^{-1}$;
5. ap-funext $xh : \text{ap}_{(-)x}(\text{funext } h) = hx$;

Proof.

1. By lemma 2.1.4 of the HoTT book, we have: $\text{refl}_u \centerdot p \centerdot \text{refl}_v = p \centerdot \text{refl}_v = p$. Because lemma is defined using path induction, we get surr-refl $\text{refl}_x \equiv \text{refl}_{\text{refl}_x}$;
2. by based path induction on p and r, we just need to define ap-trans3 $f\text{refl}_v\, q\, \text{refl}_w$. We use transitivity.

$$\begin{aligned} \text{ap}_f(\text{refl}_v \centerdot q \centerdot \text{refl}_w) &= \text{ap}_f q && \langle \text{ap}_f(\text{surr-refl } q)\rangle \\ &= \text{refl}_{fv} \centerdot \text{ap}_f q \centerdot \text{refl}_{fw} && \langle \big(\text{surr-refl } \big(\text{ap}_f q\big)\big)^{-1}\rangle \end{aligned}$$

   We get the computation rule ap-trans3 $f\ \text{refl}_x\text{refl}_x\text{refl}_x \equiv \text{refl}_{\text{refl}_{fx}}$.
3. by based path induction on p and t, we just need to define assoc5 $f\ \text{refl}_v\ q\ r\ s\ \text{refl}_y$. We again use transitivity.

$$\begin{aligned} \text{refl}_v \centerdot (q \centerdot r \centerdot s) \centerdot \text{refl}_y &= q \centerdot r \centerdot s && \langle \text{surr-refl } (q \centerdot r \centerdot s)\rangle \\ &= (\text{refl}_v \centerdot q) \centerdot r \centerdot s && \langle \text{ap}_{(-)\centerdot r \centerdot s} \text{ on HoTT 2.1.4}\rangle \\ &= (\text{refl}_v \centerdot q) \centerdot r \centerdot \big(s \centerdot \text{refl}_y\big) && \langle \text{ap}_{(\text{refl}_v \centerdot q)\centerdot r \centerdot (-)} \text{ on HoTT 2.1.4}\rangle \end{aligned}$$

   The computation rules of each rewrite step ensure assoc5 $f\ \text{refl}_x\text{refl}_x\text{refl}_x\text{refl}_x\text{refl}_x \equiv \text{refl}_{\text{refl}_x}$.
4. by based path induction on p and q. We set inv-trans $\text{refl}_v\text{refl}_v :\equiv \text{refl}_{\text{refl}_v}$;
5. Recall that happly $= \lambda p, x.\text{ap}_{(-)x} p$ ([21], lemma 3.11). So the statement is just an alternative form of propositional computation rule for identity types between functions.

□

These lemmas will work as a toolbox for the hairy path algebra lying ahead.

3.2 W-Suspension Algebras

For the rest of this chapter, we work with fixed $\mathcal{W}$-suspension signature $\mathcal{S} :\equiv (A, B, C, l, r)$. Our starting point are the initial semantics for $\mathcal{W}$-suspensions. We refer to them as "$\mathcal{W}$-suspension algebras" to remark the difference from the lawful algebras we dealt with in the previous chapter.

Definition 3.6. A *$\mathcal{W}$-suspension algebra* over $\mathcal{W}$-suspension signature $\mathcal{S}$ and on universe $\mathcal{U}_j$ is a triple consisting of:

- a carrier $D : U_j$;
- a point algebra map $d : \prod_{a:A}(Ba \to D) \to D$;
- a path algebra map $p : \prod_{c:C} \prod_{u:B(lc)\to D} \prod_{v:B(rc)\to D}(d(lc)u =_D d(rc)v)$.

We use $\mathcal{S}$– $\mathrm{WSusAlgebra}_{\mathcal{U}_j}$ to denote the type of $\mathcal{W}$-suspension algebras over $\mathcal{S}$ and on $\mathcal{U}_j$.

These algebras come with their own notion of homomorphism.

Definition 3.7. A *$\mathcal{W}$-suspension algebra homomorphism* between two $\mathcal{W}$-suspension algebras (D, d, p) and (E, e, q) over signature $\mathcal{S}$ and on universe $\mathcal{U}_j$ is a triple (f, β, θ) where:

- $f : D \to E$;
- $\beta : \prod_{a:A} \prod_{t:Ba\to D}(f(dat) =_E ea(f \circ t))$
- $\theta : \prod_{c:C} \prod_{t:B(lc)\to D} \prod_{s:B(rc)\to D}\big(\mathrm{ap}_f(pcts) = \beta(lc)t \blacksquare qc(f\circ t)(f\circ s) \blacksquare (\beta(rc)s)^{-1}\big)$.

We use $\mathcal{S}$– $\mathrm{WSusAlgHomomorphism}_{\mathcal{U}_j}(D, d, p)(E, e, q)$ to denote the type of $\mathcal{W}$-suspension algebras homomorphisms between (D, d, p) and (E, e, q).

In the definition above, β is the usual witness for algebra homomorphisms. Fixed $a : A$, we can see βa as a commutativity proof for the following diagram.

$$\begin{array}{ccc} D^{Ba} & \xrightarrow{f\circ(-)} & E^{Ba} \\ \Big\downarrow{\scriptstyle d} & \swarrow{\scriptstyle \beta a} & \Big\downarrow{\scriptstyle e} \\ D & \xrightarrow{\quad f \quad} & E \end{array}$$

Similarly, θ ensures the structure given by the path algebra map is also preserved.

$$\begin{array}{ccc} & \mathrm{ap}_f(pcts) & \\ f(d(lc)t) & \longrightarrow & f(d(rc)s) \\ \Big\downarrow{\scriptstyle \beta(lc)t} & & \Big\downarrow{\scriptstyle \beta(rc)s} \\ & qc(f\circ t)(f\circ s) & \\ e(lc)(f\circ t) & \longrightarrow & e(rc)(f\circ s) \end{array}$$

As per usual, algebras in initial semantics model recursive definitions. Similarly, an inductive construction/proof is represented using fibered algebras.

Definition 3.8. Let (D, d, p) be a $\mathcal{W}$-suspension algebra over $\mathcal{W}$-suspension signature $\mathcal{S}$ on universe $\mathcal{U}_j$. A *fibered $\mathcal{W}$-suspension algebra* over (D, d, p) is a triple consisting of:

- a fibered carrier $E : D \to U_j$;
- a fibered point algebra map $e : \prod_{a:A} \prod_{t:Ba \to D} \left(\prod_{b:Ba} E(tb) \right) \to E(dat)$;
- a fibered path algebra map
  $q : \prod_{c:C} \prod_{t:B(lc) \to D} \prod_{s:B(rc) \to D} \prod_{u:\prod_{b:B(lc)} E(tb)} \prod_{v:\prod_{b:B(rc)} E(sb)} (\mathrm{transport}^E(pcts)(e(lc)tu) =_{E(d(rc)s)e(rc)sv}$.

We use $\mathcal{S}$– $\mathrm{FibWSusAlgebra}_{\mathcal{U}_j}^{(D,d,p)}$ to denote the type of fibered $\mathcal{W}$-suspension algebras over $\mathcal{S}$ and on $\mathcal{U}_j$.

The corresponding notion of homomorphism is the following.

Definition 3.9. A *fibered $\mathcal{W}$-suspension algebra homomorphism* from $\mathcal{W}$-suspension algebra (D, d, p) over $\mathcal{S}$ and on $\mathcal{U}_j$ to fibered $\mathcal{W}$-suspension algebra (E, e, q) over (D, d, p) is a triple (f, β, θ) where:

- $f : \prod_{x:D} E$;
- $\beta : \prod_{a:A} \prod_{t:Ba \to D} (f(dat) =_E ea(f \circ t))$
- $\theta :$
  $\prod_{c:C} \prod_{t:B(lc) \to D} \prod_{s:B(rc) \to D} \left(\mathrm{apd}_f(pcts) = \mathrm{ap}_{\mathrm{transport}^E(pcts)}(\beta(lc)t) \cdot qcts(f \circ t)(f \circ s) \cdot (\beta(rc)s)^{-1} \right)$.

We use $\mathcal{S}$– $\mathrm{WSusAlgHomomorphism}_{\mathcal{U}_j}(D, d, p)(E, e, q)$ to denote the type of $\mathcal{W}$-suspension algebras homomorphisms between (D, d, p) and (E, e, q).

The diagram for β is unchanged, while the one for θ is slightly modified.

$$\begin{array}{ccc}
\mathrm{transport}^E(pcts)(f(d(lc)t)) & \xrightarrow{\mathrm{apd}_f(pcts)} & f(d(rc)s) \\
\Big| & & \Big| \\
\Big| \mathrm{ap}_{\mathrm{transport}^E(pcts)}\beta(lc)t & & \Big| \beta(rc)s \\
\Big| & & \Big| \\
\mathrm{transport}^E(pcts)(e(lc)t(f \circ t)) & \xrightarrow[qcts(f \circ t)(f \circ s)]{} & e(rc)s(f \circ s)
\end{array}$$

If we fix a universe $\mathcal{U}_j$ to work in, homotopy initiality in $\mathcal{U}_j$ is equivalent to satisfying the induction principle for $\mathcal{W}$-suspensions, as shown by K. Sojakova [24][6].

Theorem 3.10. Let $\mathcal{X}$ be a $\mathcal{W}$-suspension algebra on some universe $\mathcal{U}_j$. Then

$$\prod_{\mathcal{Y}:\mathcal{S}\text{–}\mathrm{FibWSusAlgebra}_{\mathcal{U}_j}} \mathcal{S}\text{–}\,\mathrm{FibWSusAlgHomomorphism}_{\mathcal{U}_j} \mathcal{X}\mathcal{Y}$$

(which is to say, the induction principle for $\mathcal{X}$) is type-theoretically equivalent to "$\mathcal{X}$ is homotopy initial among $\mathcal{W}$-suspension algebras".

[6]This result was originally used to identify a universal property characterising a propositional variant of $\mathcal{W}$-suspensions. We, on the other hand, are encoding "proper" $\mathcal{W}$-suspensions, and will only use this equivalence to ensure the η-principle holds for said encodings.

As anticipated, we will leave "homotopy" implicit when talking about (weak) initiality. Next, we give the type of our (non-fibered) algebras the structure of a wild category.

Theorem 3.11. $\mathcal{W}$-suspension algebras on some universe $\mathcal{U}_j$ and the $\mathcal{W}$-suspension algebra homomorphisms between them form a wild category, $\mathcal{S}\text{–}\,\mathrm{WSusAlg}_{\mathcal{U}_j}$.

Proof. We start by defining the identity morphism.

$$1_{(D,d,p)} :\equiv (\mathrm{id}_D,$$
$$\lambda a : A, t : Ba \to D.\mathrm{refl}_{dat},$$
$$\lambda c : C, t : B(lc) \to D, s : B(rc) \to D.\ \text{ap-id}\ (pcts) \centerdot (\text{surr-refl}\ (pcts))^{-1})$$

Pictorially, the third component would be

$$\mathrm{ap}_{\mathrm{id}_D}(pcts) \overline{}^{\text{ap-id}\ (pcts)} pcts \overline{}^{(\text{surr-refl}\ (pcts))^{-1}} \mathrm{refl}_{d(lc)t} \centerdot pcts \centerdot \mathrm{refl}_{d(rc)s}$$

Morphism composition is more complex. Given

$$(D, d, p) \xrightarrow{f,\beta,\theta} (E, e, q) \xrightarrow{g,\gamma,\iota} (H, h, o)$$

we define

$$(g, \gamma, \iota) \circ (f, \beta, \theta) :\equiv \big(g \circ f,$$
$$\lambda a : A, t : Ba \to D.\mathrm{ap}_g(\beta at) \centerdot \gamma a(f \circ t),$$
$$\ldots\big)$$

The second component arises from the following diagram.

$$\begin{array}{ccccc} D^{Ba} & \xrightarrow{f\circ(-)} & E^{Ba} & \xrightarrow{g\circ(-)} & H^{Ba} \\ \downarrow{\scriptstyle d} & \beta a & \downarrow{\scriptstyle e} & \gamma a & \downarrow{\scriptstyle h} \\ D & \xrightarrow{f} & E & \xrightarrow{g} & H \end{array}$$

We present the third component of the composition as the following chain of equalities

$$\begin{aligned}
\mathrm{ap}_{g\circ f}(pcts) &= && \langle(\text{ap-comp}\ gf(pcts))^{-1}\rangle \\
\mathrm{ap}_g\big(\mathrm{ap}_f(pcts)\big) &= && \langle \mathrm{ap}_{\mathrm{ap}_g}(\theta cts)\rangle \\
\mathrm{ap}_g(\beta(lc)t \centerdot qc(f\circ t)(f\circ s) \centerdot (\beta(rc)s)^{-1}) &= && \langle\text{ap-trans3}\ g(\beta(lc)t)(qc(f\circ t)(f\circ s))(\beta(rc)s)^{-1}\rangle \\
\mathrm{ap}_g(\beta(lc)t) \centerdot \mathrm{ap}_g(qc(f\circ t)(f\circ s)) \centerdot \mathrm{ap}_g(\beta(rc)s)^{-1} &= && \langle \mathrm{ap}_{\mathrm{ap}_g(\beta(lc)t)\centerdot(-)\centerdot \mathrm{ap}_g(\beta(rc)s)^{-1}}(\iota c(f\circ t)(f\circ s))\rangle \\
\mathrm{ap}_g(\beta(lc)t) \centerdot (\gamma(lc)(f\circ t) \circ oc(g\circ f\circ t)(g\circ f\circ s) \circ (\gamma(rc)(f\circ s))^{-1}) \centerdot \mathrm{ap}_g(\beta(rc)s)^{-1} &= && \langle \text{help}\ (f,\beta)(g,\gamma)ts\rangle \\
\big(\mathrm{ap}_g(\beta(lc)t) \centerdot \gamma(lc)(f\circ t)\big) \circ oc(g\circ f\circ t)(g\circ f\circ s) \circ \big(\mathrm{ap}_g(\beta(rc)s) \centerdot \gamma(rc)(f\circ s)\big)^{-1}, &&&
\end{aligned}$$

where

$$\begin{aligned}\text{help}\ (f,\beta)(g,\gamma)ts :\equiv\ & \text{assoc5}\ \big(\text{ap}_g(\beta(lc)t)\big)(\gamma(lc)(f\circ t))(oc(g\circ f\circ t)(g\circ f\circ s))(\gamma(rc)(f\circ s))^{-1}\big(\text{ap}_g(\beta(rc)s)^{-1}\big)\\ & \blacksquare\ \text{ap}_{(\text{ap}_g(\beta(lc)t)\,\blacksquare\,\gamma(lc)(f\circ t))\,\blacksquare\,(oc(g\circ f\circ t)(g\circ f\circ s))\,\blacksquare\,((\gamma(rc)(f\circ s))^{-1}\,\blacksquare\,(-))}(\text{ap-inv}\ g(\beta(rc)s))\\ & \blacksquare\ \text{ap}_{(\text{ap}_g(\beta(lc)t)\,\blacksquare\,\gamma(lc)(f\circ t))\,\blacksquare\,(oc(g\circ f\circ t)(g\circ f\circ s))\,\blacksquare\,(-)}\big(\text{inv-trans}\ \big(\text{ap}_g(\beta(rc)s)\big)(\gamma(rc)(f\circ s))\big)\big)^{-1}.\end{aligned}$$

This concludes the constructions of the identity and composition operations. We will sometimes use $(\gamma\circ\beta)$ or $(\iota\circ\theta)$ to refer to the second and third component of $(g,\gamma,\iota)\circ(f,\beta,\theta)$ respectively, when the rest of the data is clear from the context. We now prove the axioms of a (non-univalent) category, starting from left unitality, i.e. $1_{(E,e,q)}\circ(f,\beta,\theta)=(f,\beta,\theta)$ for any $(f,\beta,\theta):(D,d,p)\to(E,e,q)$. We do this componentwise, using $\text{pair}^=$ twice. Since $\text{id}_E\circ f\equiv f$, we can ignore transport when comparing the second components. We show they are equal by applying function extensionality twice and introducing $a:A,t:Ba\to D$. We get

$$\text{ap}_{\text{id}_E}(\beta at)\blacksquare\text{refl}_{ea(f\circ t)}\overset{\text{reflR}\ \left(\text{ap}_{\text{id}_E}\beta at\right)}{\longrightarrow}\text{ap}_{\text{id}_E}(\beta at)\overset{\text{ap-id}\ (\beta at)}{\longrightarrow}\beta at$$

where reflR witnesses unitality of refl on the right. To compare the third components, we need to transfer along this proof obtained by double function extensionality. We use function extensionality three times, fixing $c:C,t:B(lc)\to D,s:B(rc)\to D$. We can use tr-fun $((-)cts)$ to bring c, t, and s inside the transport, and theorem 2.11.3 from the HoTT book to get rid of the latter. Unfolding our definition of composition (help included) we get a sequence of 11 paths that we need to show to be equal to θcts. The first is equal to $\text{refl}_{\text{ap}_f(pcts)}$, because it is the action of a constant function on a path. The last 6 can also be shown to be equal to reflexivity. This is done by generalising $\beta(lc)t$ and $\beta(rc)s$ to arbitrary based paths, and then performing based paths induction on the latter. All the remaining paths can be rewritten as $\text{ap}_x(\text{funext ap} - \text{id})$ for some x (with the help of ap-funext), or the inverse of something in this form, so by using theorem 2.11.3 again (twice), we get to

$$(\text{funext ap-id})_*\big((\text{funext ap-id})_*^{-1}(\theta cts)\big)$$

which simplifies to the desired result. We now move to right unitality, by showing $(f,\beta,\theta)\circ 1_{(D,d,p)}=(f,\beta,\theta)$. Again, we do this componentwise, using $\text{pair}^=$ twice. Since $f\circ\text{id}_D\equiv f$, we ignore transport when comparing the second components. We apply function extensionality twice introducing $a:A,t:Ba\to D$. We get

$$\text{ap}_f(\text{refl}_{dat})\blacksquare\beta at\equiv\text{refl}_{f(dat)}\blacksquare\beta at\overset{\text{reflL}\ (\beta at)}{\longrightarrow}\beta at$$

where reflL witnesses unitality of refl on the left. To compare the third components, we need to transfer along this proof obtained by double function extensionality. We use function extensionality three times, fixing $c:C,t:B(lc)\to D,s:B(rc)\to D$. Again, we use tr-fun $((-)cts)$ to bring c, t, and s inside the transport, and theorem 2.11.3 of the HoTT book to get rid of said transport. This time, we unfold our definitions of morphism composition, help, ap-trans3, and assoc5. Much like before, the first path of the resulting sequence is equal to reflexivity, since it is the action of a constant function on a path. After that, all the paths up to halwfay through the unfolded definition of ap-trans3 form a loop on $\text{ap}_f(pcts)$. This can also be easily shown to be equal to reflexivity. Similarly, the end of the chain is also a loop on $\beta(lc)t\blacksquare qc(f\circ t)(f\circ s)\blacksquare(\beta(rc)s)^{-1}$. It is again shown to be equal to reflexivity by generalising $\beta(lc)t$ and $(\beta(rc)s)^{-1}$ to generic based paths and then

performing double based path induction on both. What's left of the original chain is propositionally equal to

$$(\text{funext surr-refl})_*\big((\text{funext surr-refl})_*^{-1}(\theta cts)\big)$$

and this is just θcts as desired. Only associativity is now left to prove. We consider a sequence of morphisms like the following

$$(D,d,p) \overset{(f,\beta,\theta)}{\longrightarrow} (E,e,q) \overset{(g,\gamma,\iota)}{\longrightarrow} (I,i,o) \overset{(h,\lambda,\kappa)}{\longrightarrow} (J,j,n)$$

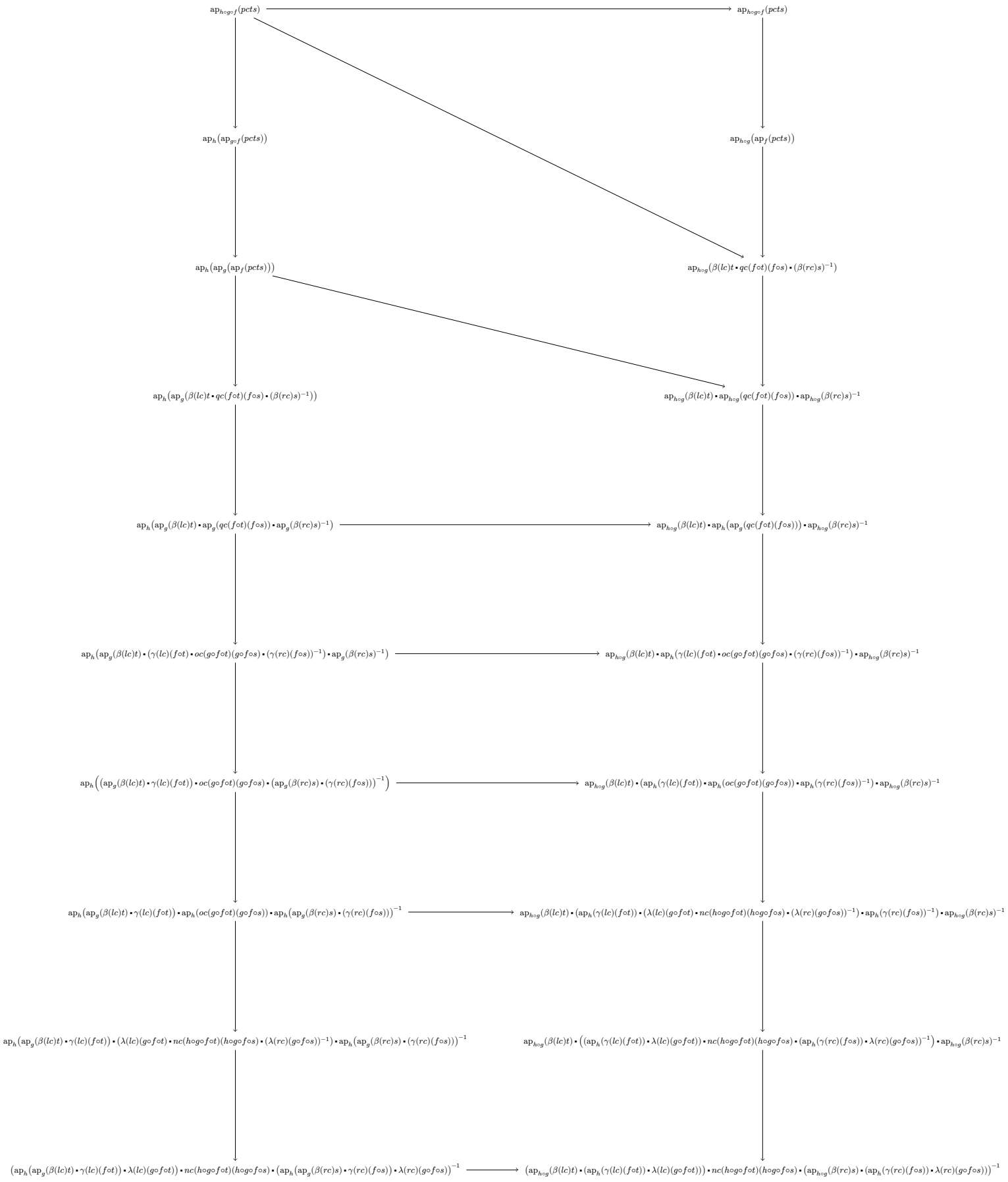
and prove

$$(h,\lambda,\kappa)\circ((g,\gamma,\iota)\circ(f,\beta,\theta)) = ((h,\lambda,\kappa)\circ(g,\gamma,\iota))\circ(f,\beta,\theta).$$

We need to use $\text{pair}^=$ twice again to compare the two morphisms componentwise. The equality between the first components holds definitionally, so we ignore transport when comparing the second ones. By function extensionality, let $a : A, t : Ba \to D$.

$$\begin{aligned}
\text{ap}_h\big(\text{ap}_g(\beta at)\circ\gamma at\big)\circ\lambda at &= \big(\text{ap}_h\big(\text{ap}_g(\beta at)\big)\circ\text{ap}_h(\gamma at)\big)\circ\lambda at && \langle \text{ap}_{(-)\circ\lambda at}\big(\text{ap-trans } h\big(\text{ap}_g(\beta at)\big)(\gamma at)\big)\rangle \\
&= \big(\text{ap}_{h\circ g}(\beta at)\circ\text{ap}_h(\gamma at)\big)\circ\lambda at && \langle \text{ap}_{((-)\circ\text{ap}_h(\gamma at))\circ\lambda at}(\text{ap-comp } hg(\beta at))\rangle \\
&= \text{ap}_{h\circ g}(\beta at)\circ(\text{ap}_h(\gamma at)\circ\lambda at) && \langle \text{trans-assoc } \big(\text{ap}_{h\circ g}(\beta at)\big)(\text{ap}_h(\gamma at))(\lambda at)\rangle
\end{aligned}$$

Above, trans-assoc is just associativity of paths composition. The only remaining equality check to do is the one between the third components, and we need transport along the proof just shown to make this typecheck. As per usual, we employ function extensionality to fix $c : C, t : B(lc) \to D, s : B(rc) \to D$. Then, we take advantage of tr-fun $((-)cts)$ to bring these three variables inside the transport, which can then be rewritten using theorem 2.11.3. By using ap-funext, we can cancel out the resulting ap operator and the funext from the original equality proof between the second components. The resulting is the following diagram. The top and bottom side are the result of eliminating transport using theorem 2.11.3 from the HoTT book. The left (right) side is obtained by unfolding the left (right) side of the equation. Our goal is hence commutativity for the outer perimeter. This is shown by breaking it down in much smaller polygons, each of which commutes.



So all the category axioms hold. □

Because of the impredicativity of our bottom universe $\mathcal{U}_0$, achieving weak initiality is very easy. From now on, we don't need to specify the universe we work in any more, as it is always $\mathcal{U}_0$.

Proposition 3.12. The wild category $\mathcal{S}$– WSusAlg has a weakly initial object $(W_0, \mathrm{sup}_0, \mathrm{eq}_0)$.

Proof. We define:

$$\begin{aligned} W_0 &:\equiv \prod_{(D,d,p):\mathcal{S}\text{–WSusAlgebra}} D \\ \mathrm{sup}_0 &:\equiv \lambda a : A, t : Ba \to W_0, (D,d,p) : \mathcal{S}\text{– WSusAlgebra } .da(((-)(D,d,p))\circ t) \\ \mathrm{eq}_0 &:\equiv \lambda c : C, u : B(lc) \to W_0, v : B(rc) \to W_0.\ \text{funext} \\ &\quad \lambda(D,d,p).pc((-)(D,d,p))\circ u)(((-)(D,d,p))\circ v) \end{aligned}$$

Given a second $\mathcal{W}$-suspension algebra over $\mathcal{S}$, say (D,d,p), we have a function

$$\begin{aligned} \mathrm{rec}_0(D,d,p) &: W_0 \to D \\ \mathrm{rec}_0(D,d,p)\alpha &:\equiv \alpha(D,d,p) \end{aligned}$$

between the two carriers. This extends to a proper algebra morpishm as witnessed by

$$\lambda a : A, t : Ba \to W_0.\mathrm{refl}_{da(((-)(D,d,p))\circ t)}.$$

Finally, we add the $\mathcal{W}$-suspension algebra homomorphism structure by proving:

$$\begin{aligned} &\mathrm{ap}_{(-)(D,d,p)}(\mathrm{eq}_0 cts) \\ &\equiv \mathrm{ap}_{(-)(D,d,p)}(\text{funext } \lambda\mathcal{X}.qc(((-)\mathcal{X})\circ t)(((-)\mathcal{X})\circ s)) \\ &= pc((-)(D,d,p))\circ t)(((-)(D,d,p))\circ s) && \langle \text{ap-funext } (D,d,p)(\lambda\mathcal{X}.qc(((-)\mathcal{X})\circ t)(((-)\mathcal{X})\circ s))\rangle \\ &= \mathrm{refl} \blacksquare pc((-)(D,d,p))\circ t)(((-)(D,d,p))\circ s) \blacksquare \mathrm{refl} && \langle \text{surr-refl } (pc((-)(D,d,p))\circ t)(((-)(D,d,p))\circ s))\rangle \end{aligned}$$

□

3.3 Naturality

The first attempt at a proposed refinement simply encodes the induction property we would like our encodings to satisfy.

Definition 3.13. Let (D,d,p) be a $\mathcal{W}$-suspension algebra over $\mathcal{W}$-suspension signature $\mathcal{S}$. The *inductivity of* (D,d,p), or $\mathrm{Ind}_{(D,d,p)}$, is defined as the following type family over D.

$$\begin{aligned} \mathrm{Ind}_{(D,d,p)} &: D \to \mathcal{U}_0 \\ \mathrm{Ind}_{(D,d,p)}\alpha &:\equiv \prod_{(E,e,q):\mathcal{S}\text{–FibWSusAlgebra}^{(D,d,p)}} E\alpha \end{aligned}$$

The definition above is expressing the idea that proofs by induction performed on our signature $\mathcal{S}$ can be used to instance a concrete proof of the predicate in question for α. So the obvious next step will be restricting our naïve encoding to terms equipped with an inductivity proof. For starters, we observe that the $\mathcal{W}$-suspension algebra structure is always preserved when refining the carrier of an arbitrary $\mathcal{W}$-suspension algebra structure this way.

Proposition 3.14. Let (D, d, p) be a $\mathcal{W}$-suspension algebra over $\mathcal{W}$-suspension algebra signature $\mathcal{S}$. Then, $\sum_{\alpha:D} \mathrm{Ind}_{(D,d,p)}\alpha$ can be given the structure of a $\mathcal{W}$-suspension algebra over $\mathcal{W}$-suspension signature $\mathcal{S}$, too.

Proof. As a point algebra map, we take

$$d' : \prod_{a:A}\left(Ba \to \sum_{\alpha:D} \mathrm{Ind}_{(D,d,p)}\alpha\right) \to \sum_{\alpha:D} \mathrm{Ind}_{(D,d,p)}\alpha$$

$$d'at :\equiv \big(dat, \lambda(E,e,q) : \mathcal{S}\text{–}\,\mathrm{FibWSusAlgebra}^{(D,d,p)}.ea(\mathrm{pr}_1 \circ t)((-)(E,e,q) \circ \mathrm{pr}_2 \circ t)\big).$$

As a path algebra map, we take

$$p' : \prod_{c:C} \quad \prod_{u:B(lc)\to \sum_{\alpha:D} \mathrm{Ind}_{(D,d,p)}\alpha} \quad \prod_{v:B(rc)\to \sum_{\alpha:D} \mathrm{Ind}_{(D,d,p)}\alpha} \Big(d(lc)u =_{\sum_{\alpha:D} \mathrm{Ind}_{(D,d,p)}\alpha} d(rc)v\Big)$$

$$\begin{aligned} p'cuv :\equiv\ & \mathrm{pair}^{=}(pc(\mathrm{pr}_1 \circ u)(\mathrm{pr}_1 \circ v))\big(\lambda(E,e,q) : \mathcal{S}\text{–}\,\mathrm{FibWSusAlgebra}^{(D,d,p)}. \\ & \text{tr-fun } (\_ \mapsto (-)(E,e,q))(pc(\mathrm{pr}_1 \circ u)(\mathrm{pr}_1 \circ v))(...) \centerdot \\ & qc(\mathrm{pr}_1 \circ u)(\mathrm{pr}_1 \circ v)(((-)(E,e,q)) \circ \mathrm{pr}_2 \circ u)(((-)(E,e,q)) \circ \mathrm{pr}_2 \circ v)\big). \end{aligned}$$

where we use the underscore character (_) to bind unused variables. The ellipsis ("...") replaces the term we are transporting. □

We denote the n-th iterated refinement[7] of $(W_0, \mathrm{sup}_0, \mathrm{eq}_0)$ via Proposition 3.14 using the tuple $(W_n, \mathrm{sup}_n, \mathrm{eq}_n)$.

Proposition 3.15. Fixed $n : \mathbb{N}$, the two inclusion maps $\pi_n : W_{Sn} \to W_n$ and $\rho_n : W_n \to W_0$ can be given a $\mathcal{W}$-suspension algebra homomorphism structure between the respective $\mathcal{W}$-suspension algebras.

Proof. To show that the point constructors are preserved, we can use pointwise reflexivity in both cases. So we only need to show that the path constructors are preserved. For π_n:

$$\begin{aligned} \mathrm{ap}_{\pi_n}(\mathrm{eq}_{Sn}cts) &\equiv \mathrm{ap}_{\mathrm{pr}_1}(\mathrm{pair}^{=}(\mathrm{eq}_n c(\mathrm{pr}_1 \circ t)(\mathrm{pr}_1 \circ s))(...)) \\ &= \mathrm{eq}_n c(\mathrm{pr}_1 \circ t)(\mathrm{pr}_1 \circ s) && \langle \text{HoTT book, 2.7.2} \rangle \\ &= \mathrm{refl}_{\mathrm{sup}_n(lc)(\mathrm{pr}_1 \circ t)} \centerdot \mathrm{eq}_n c(\mathrm{pr}_1 \circ t)(\mathrm{pr}_1 \circ s) \centerdot \mathrm{refl}_{\mathrm{sup}_n(rc)(\mathrm{pr}_1 \circ s)} && \langle (\text{surr-refl } (\mathrm{eq}_n c(\mathrm{pr}_1 \circ t)(\mathrm{pr}_1 \circ s)))^{-1} \rangle \\ &\equiv \mathrm{refl}_{\mathrm{sup}_n(lc)(\pi_1 \circ t)} \centerdot \mathrm{eq}_n c(\pi_1 \circ t)(\pi_1 \circ s) \centerdot \mathrm{refl}_{\mathrm{sup}_n(rc)(\pi_1 \circ s)}. \end{aligned}$$

For ρ_n, by induction:

- if $n = 0$, then ρ_n can be extended to the identity morphism;
- if $n = m + 1$, then we can define ρ_n as the morphism obtained composing ρ_m after π_m.

□

Corollary 3.16. Given $n : \mathbb{N}$ and a $\mathcal{W}$-suspension algebra (D, d, p) over $\mathcal{W}$-suspension signature $\mathcal{S}$, we have a $\mathcal{W}$-suspension algebra homomorphism $\mathrm{rec}_{Sn}(D, d, p)$ from $(W_{Sn}, \mathrm{sup}_{Sn}, \mathrm{eq}_{Sn})$ to (D, d, p)

[7] We are making use of the newly assumed natural numbers type here just for ease of exposition. In fact, we will never need anything more than the two-iterations refinement. Assuming $\mathbb{N}$ will actually be made necessary by the next section.

defined by composing $\mathrm{rec}_0(D,d,p)$ after ρ_{Sn} in $\mathcal{S}$– WSusAlg, where S is the "successor" constructor for $\mathbb{N}$. So $(W_1, \mathrm{sup}_1, \mathrm{eq}_1)$ is weakly initial.

The first layer of inductivity is already enough to recover a condition that corresponds to the naturality property from the previous chapter. Of course, in contrast with the set-truncated setting, in general we are not dealing with a mere proposition any more.

Lemma 3.17. For all $(f,\beta,\theta) : \mathcal{S}$– WSusAlgHomomorphism $(D,d,p)(E,e,q)$, we have

$$(f,\beta,\theta)\circ(\mathrm{rec}_1(D,d,p),(\_\mapsto\_\mapsto\mathrm{refl}),...) = (\mathrm{rec}_1(E,e,q),(\_\mapsto\_\mapsto\mathrm{refl}),...)$$

where the two ellipses replace the proof (found in Proposition 3.12) that $\mathrm{rec}_0(D,d,p)$ $(\mathrm{rec}_0(E,e,q))$ extends from an algebra morphism to a $\mathcal{W}$-suspension algebra morphism.

Proof. We use $\mathrm{pair}^=$ twice to compare the two morphisms componentwise. To check the underlying functions for equality, we use function extensionality. Let $\alpha : W_1$. We have to show $C(\mathrm{pr}_1\alpha)$, with

$$Cx :\equiv f(x(D,d,p)) =_E x(E,e,q),$$

which $\mathrm{pr}_2\alpha$ allows us to prove by induction. We set $\mathscr{d} :\equiv \mathrm{rec}_0(D,d,p)$ and $\mathscr{e} :\equiv \mathrm{rec}_0(E,e,q)$:

- for the point constructors, let $a : A, t : Ba \to W_0, g : \prod_{b:Ba} C(tb)$. We have:

$$\begin{aligned} f(\mathrm{sup}_0 at(D,d,p)) &\equiv f(da(\mathscr{d}\circ t)) \\ &= ea(f\circ\mathscr{d}\circ t) && \langle \beta a(\mathscr{d}\circ t)\rangle \\ &= ea(\mathscr{e}\circ t) && \langle \mathrm{ap}_{ea((-)\circ t)}(\mathrm{funext}\ g)\rangle \\ &\equiv \mathrm{sup}_0 at(E,e,q) \end{aligned}$$

- for the path constructors, let $c : C, t : B(lc) \to W_0, s : B(rc) \to W_0, u : \prod_{b:B(lc)} C(tb), v : \prod_{b:B(rc)} C(sb)$.

$$\begin{aligned} &\mathrm{transport}^C(\mathrm{eq}_0 cts)\big(\beta(lc)(\mathscr{d}\circ t)\centerdot \mathrm{ap}_{e(lc)((-)\circ t)}(\mathrm{funext}\ u)\big) \\ &= \mathrm{ap}_{f\circ\mathscr{d}}(\mathrm{eq}_0 cts)^{-1}\centerdot\beta(lc)(\mathscr{d}\circ t)\centerdot \mathrm{ap}_{e(lc)((-)\circ t)}(\mathrm{funext}\ u)\centerdot \mathrm{ap}_{\mathscr{e}}(\mathrm{eq}_0 cts) && \langle \text{HoTT book, 2.11.3}\rangle \\ &= \beta(rc)(\mathscr{d}\circ s)\centerdot(qc(f\circ\mathscr{d}\circ t)(f\circ\mathscr{d}\circ s))^{-1}\centerdot \mathrm{ap}_{e(lc)((-)\circ t)}(\mathrm{funext}\ u)\centerdot \mathrm{ap}_{\mathscr{e}}(\mathrm{eq}_0 cts) && \langle \text{morphism}\rangle \\ &= \beta(rc)(\mathscr{d}\circ s)\centerdot(qc(f\circ\mathscr{d}\circ t)(f\circ\mathscr{d}\circ s))^{-1}\centerdot \mathrm{ap}_{e(lc)((-)\circ t)}(\mathrm{funext}\ u)\centerdot qc(\mathscr{e}\circ t)(\mathscr{e}\circ s) && \langle \text{morphism}\rangle \\ &= \beta(rc)(\mathscr{d}\circ s)\centerdot(qc(\mathscr{e}\circ t)(f\circ\mathscr{d}\circ s))^{-1}\centerdot qc(\mathscr{e}\circ t)(\mathscr{e}\circ s) && \langle \text{apd}\rangle \\ &= \beta(rc)(\mathscr{d}\circ s)\centerdot \mathrm{ap}_{ea(rc)}(\mathrm{funext}\ v). && \langle \text{apd}\rangle \end{aligned}$$

So our proof by induction is complete. We name the corresponding fibered W-suspenison algebra $\mathcal{C}$ and set $\mathscr{c} :\equiv (-)\mathcal{C}$. We now transport along this prior proof term, that we call ❦, obtained using function extensionality and an inductivity witness, to be able to state the equality between the second components. We use function extensionality again, fixing $a : A, t : Ba \to W_1$, so that we only have to prove

$$❦_*\big(\lambda a : A, t : Ba \to W_1.\mathrm{refl}_{f(da(\mathscr{d}\circ t))}\circ\beta a(\mathscr{d}\circ t)\big)at = \mathrm{refl}_{ea(\mathscr{e}\circ t)}.$$

Indeed, the following equalities hold.

$$
\begin{aligned}
&\text{❦}_*\Big(\lambda a : A, t : Ba \to W_1.\mathrm{refl}_{f(da(\mathscr{d}\circ t))}\circ\beta a(\mathscr{d}\circ t)\Big)at \\
&= \text{❦}_*(\lambda a : A, t : Ba \to W_1.\beta a(\mathscr{d}\circ t))at && \langle \mathrm{ap}_{\text{❦}_*(\lambda a,t.(-))at}(\mathrm{reflL}\ (\beta a(\mathscr{d}\circ t)))\rangle \\
&= \text{❦}_*(\beta a(\mathscr{d}\circ t)) && \langle \text{tr-fun } ((-)at)\text{❦}\lambda a,t.\beta a(\mathscr{d}\circ t)\rangle \\
&= \Big(\mathrm{ap}_{(-)(\mathrm{sup}_1 at)}\text{❦}\Big)^{-1} \cdot \beta a(\mathscr{d}\circ t) \cdot \mathrm{ap}_{ea((-)\circ t)}\text{❦} && \langle \text{HoTT book, 2.11.3}\rangle \\
&= (\beta a(\mathscr{d}\circ t) \cdot \mathrm{ap}_{ea}(\text{funext } (\mathscr{c}\circ\mathrm{pr}_2\circ t)))^{-1} \cdot \beta a(\mathscr{d}\circ t) \cdot \mathrm{ap}_{ea((-)\circ t)}\text{❦} && \langle \mathrm{ap}_{(-)^{-1}\cdot\beta a(\mathscr{d}\circ t)\cdot\mathrm{ap}_{ea((-)\circ t)}\text{❦}}(\text{ap-funext } (\mathrm{sup}_1 at)(...))\rangle \\
&= (\beta a(\mathscr{d}\circ t) \cdot \mathrm{ap}_{ea}(\text{funext } (\mathscr{c}\circ\mathrm{pr}_2\circ t)))^{-1} \cdot \beta a(\mathscr{d}\circ t) \cdot \mathrm{ap}_{ea}\Big(\mathrm{ap}_{(-)\circ t}\text{❦}\Big) && \langle \mathrm{ap}_{\dots\cdot(-)}(\text{ap-comp } (ea)((-)\circ t)\text{❦})^{-1}\rangle \\
&= (\beta a(\mathscr{d}\circ t) \cdot \mathrm{ap}_{ea}(\text{funext } (\mathscr{c}\circ\mathrm{pr}_2\circ t)))^{-1} \cdot \beta a(\mathscr{d}\circ t) \cdot \mathrm{ap}_{ea}(\text{funext } (\mathscr{c}\circ\mathrm{pr}_2\circ t)) && \langle \mathrm{ap}_{\dots\cdot\mathrm{ap}_{ea}(-)}(\text{ap-funext-precomp } t(\mathscr{c}\circ\mathrm{pr}_2))\rangle \\
&= \mathrm{refl}_{ea(\mathscr{e}\circ t)} && \langle \text{trans-invL } (\beta a(\mathscr{d}\circ t) \cdot \mathrm{ap}_{ea}(\text{funext } (\mathscr{c}\circ\mathrm{pr}_2\circ t)))\rangle
\end{aligned}
$$

where ap-funext-precomp $eh : \mathrm{ap}_{(-)\circ e}(\text{funext } h) = \text{funext } (h\circ e)$ is easily proven and trans-invL is the left inverse path property. We denote this proof of equality between the second components, that we have constructed by function extensionality, ♭. We are now left with comparing the third components. To state their equality, we need to transport along $\text{pair}^=\text{❦}\flat$. As many times before, we can use function extensionality to append some fixed $c : C, t : B(lc) \to W_1, s : B(rc) \to W_1$ to both sides of the equation. Then, tr-fun $((-)cts)$ allows us to bring these three newly introduced variables inside the transport on the left side. This time around, to characterise transport we need the dependent version of theorem 2.11.3 of the HoTT book, i.e. theorem 2.11.4. Our goal is now making the following diagram commmute, where ω_f stands for the third component of the $\mathcal{W}$-suspension algebra homomorpism constructed from f.

$$
\begin{array}{ccc}
(\text{pair}^=\text{❦}\flat)_*\Big(\mathrm{ap}_{f\circ\mathrm{rec}_1(D,d,p)}(\mathrm{eq}_1 cts)\Big) & \xrightarrow{\mathrm{ap}_{(\text{pair}^=\text{❦}\flat)_*}\left(\left(\theta\circ\omega_{\mathrm{rec}_1(D,d,p)}\right)cts\right)} & (\text{pair}^=\text{❦}\flat)_*\Big((\mathrm{refl}\cdot\beta(lc)(\mathrm{rec}_1(D,d,p)\circ t))\cdot qc(f\circ\mathrm{rec}_1(D,d,p)\circ t)(f\circ\mathrm{rec}_1(D,d,p)\circ s)\cdot(\mathrm{refl}\cdot\beta(rc)(\mathrm{rec}_1(D,d,p)\circ s))^{-1}\Big) \\
\Big\downarrow{\scriptstyle \mathrm{apd}_{(x,\_)\mapsto\mathrm{ap}_x(\mathrm{eq}_1 cts)}(\text{pair}^=\text{❦}\flat)^{-1}} & & \Big\downarrow{\scriptstyle \mathrm{apd}_{(x,y)\mapsto y(lc)t\cdot ea(x\circ t)(x\circ s)\cdot(y(rc)s)^{-1}}(\text{pair}^=\text{❦}\flat)} \\
\mathrm{ap}_{\mathrm{rec}_1(E,e,q)}(\mathrm{eq}_1 cts) & \xrightarrow{\omega_{\mathrm{rec}_1(E,e,q)}cts} & \mathrm{refl}\cdot qc(\mathrm{rec}_1(E,e,q)\circ t)(\mathrm{rec}_1(E,e,q)\circ s)\cdot\mathrm{refl}
\end{array}
$$

We only show how to simplify each side of the square until commutativity becomes a simple (although very tedious) exercise. By construction of the morphism on $\mathrm{rec}_1(E,e,q)$, it can be shown that the bottom side simplifies to the following chain of equalities:

$$
\begin{aligned}
&\mathrm{ap}_{\mathrm{rec}_1(E,e,q)}(\mathrm{eq}_1 cts) \\
&= \mathrm{ap}_{\mathscr{e}}\Big(\mathrm{ap}_{\mathrm{pr}_1}(\mathrm{eq}_1 cts)\Big) && \langle(\text{ap-comp } \mathscr{e}\mathrm{pr}_1(\mathrm{eq}_1 cts))^{-1}\rangle \\
&= \mathrm{ap}_{\mathscr{e}}(\mathrm{eq}_0 c(\mathrm{pr}_1\circ t)(\mathrm{pr}_1\circ s)) && \langle\mathrm{ap}_{\mathscr{e}}(\text{HoTT book, 2.6.2})\rangle \\
&= pc(\mathscr{e}\circ\mathrm{pr}_1\circ t)(\mathscr{e}\circ\mathrm{pr}_1\circ s) && \langle\text{ap-funext } (E,e,q)\lambda\mathcal{X}.pc(((-)\mathcal{X})\circ\mathrm{pr}_1\circ t)(((-)\mathcal{X})\circ\mathrm{pr}_1\circ s)\rangle \\
&= \mathrm{refl}\cdot pc(\mathscr{e}\circ\mathrm{pr}_1\circ t)(\mathscr{e}\circ\mathrm{pr}_1\circ s)\cdot\mathrm{refl} && \langle(\text{surr-refl } (pc(\mathscr{e}\circ\mathrm{pr}_1\circ t)(\mathscr{e}\circ\mathrm{pr}_1\circ s))^{-1}\rangle.
\end{aligned}
$$

The exact same reasoning applies to $\mathrm{rec}_1(D,d,p)$ and $\omega_{\mathrm{rec}_1(D,d,p)}$ on the top side, of course. Then, the argument of the transport can be simplified to:

$$
\begin{aligned}
&\mathrm{ap}_{f\circ \mathrm{rec}_1(D,d,p)}(\mathrm{eq}_1 cts) && \\
&= \mathrm{ap}_f\Big(\mathrm{ap}_{\mathrm{rec}_1(D,d,p)}(\mathrm{eq}_1 cts)\Big) && \langle\Big(\text{ap-comp } f(\mathrm{rec}_1(D,d,p))(\mathrm{eq}_1 cts)^{-1}\Big)\rangle \\
&= \mathrm{ap}_f\Big(\mathrm{ap}_{\mathscr{d}}\Big(\mathrm{ap}_{\mathrm{pr}_1}(\mathrm{eq}_1 cts)\Big)\Big) && \langle \mathrm{ap}_{\mathrm{ap}_f}\Big(\text{ap-comp } \mathscr{d}\mathrm{pr}_1(\mathrm{eq}_1 cts)^{-1}\Big)\rangle \\
&= \mathrm{ap}_f(\mathrm{ap}_{\mathscr{d}}(\mathrm{eq}_0 c(\mathrm{pr}_1\circ t)(\mathrm{pr}_1\circ s))) && \langle \mathrm{ap}_{\mathrm{ap}_f\circ \mathrm{ap}_{\mathscr{d}}}(\text{HoTT book, 2.6.2})\rangle \\
&= \mathrm{ap}_f(pc(\mathscr{d}\circ\mathrm{pr}_1\circ t)(\mathscr{d}\circ\mathrm{pr}_1\circ s)) && \langle \mathrm{ap}_{\mathrm{ap}_f}(\text{ap-funext } (D,d,p)\lambda\mathcal{X}.pc(((-)\mathcal{X})\circ\mathrm{pr}_1\circ t)((-)\mathcal{X})\circ\mathrm{pr}_1\circ t)\Big)\rangle \\
&= \beta(lc)(\mathscr{d}\circ\mathrm{pr}_1\circ t)\centerdot qc(f\circ\mathscr{d}\circ\mathrm{pr}_1\circ t)(f\circ\mathscr{d}\circ\mathrm{pr}_1\circ s)\centerdot(\beta(rc)(\mathscr{d}\circ\mathrm{pr}_1\circ s))^{-1} && \langle \theta c(\mathscr{d}\circ\mathrm{pr}_1\circ t)(\mathscr{d}\circ\mathrm{pr}_1\circ s)\rangle \\
&= (\mathrm{refl}\centerdot\beta(lc)(\mathscr{d}\circ\mathrm{pr}_1\circ t))\centerdot qc(f\circ\mathscr{d}\circ\mathrm{pr}_1\circ t)(f\circ\mathscr{d}\circ\mathrm{pr}_1\circ s)\centerdot(\beta(rc)(\mathscr{d}\circ\mathrm{pr}_1\circ s))^{-1} && \langle \mathrm{ap}_{(-)\centerdot\ldots}(\mathrm{reflL}\ (\beta(lc)(\mathscr{d}\circ\mathrm{pr}_1\circ t)))^{-1}\rangle \\
&= (\mathrm{refl}\centerdot\beta(lc)(\mathscr{d}\circ\mathrm{pr}_1\circ t))\centerdot qc(f\circ\mathscr{d}\circ\mathrm{pr}_1\circ t)(f\circ\mathscr{d}\circ\mathrm{pr}_1\circ s)\centerdot(\mathrm{refl}\centerdot\beta(rc)(\mathscr{d}\circ\mathrm{pr}_1\circ s))^{-1} && \langle \mathrm{ap}_{\ldots\centerdot(-)^{-1}}(\mathrm{reflL}\ (\beta(rc)(\mathscr{d}\circ\mathrm{pr}_1\circ s)))^{-1}\rangle
\end{aligned}
$$

As for the left side, we observe that the second component of the equality is currently unused. We can get relegate it to the first step of an equality chain by rewriting the left side to what follows:

$$
\begin{aligned}
&\mathrm{transport}^{(x,\_)\mapsto \mathrm{ap}_x(\mathrm{eq}_1 cts)}(\mathrm{pair}^{=}\text{❦}\flat)\Big(\mathrm{ap}_{f\circ\mathrm{rec}_1(D,d,p)}(\mathrm{eq}_1 cts)\Big) && \\
&= \mathrm{transport}^{\mathrm{ap}_{(-)}(\mathrm{eq}_1 cts)}\Big(\mathrm{ap}_{\mathrm{pr}_1}(\mathrm{pair}^{=}\text{❦}\flat)\Big)\Big(\mathrm{ap}_{f\circ\mathrm{rec}_1(D,d,p)}(\mathrm{eq}_1 cts)\Big) && \langle \text{tr-comp } \mathrm{pr}_1(\mathrm{pair}^{=}\text{❦}\flat)\Big(\mathrm{ap}_{f\circ\mathrm{rec}_1(D,d,p)}(\mathrm{eq}_1 cts)\Big)\rangle \\
&= \mathrm{transport}^{\mathrm{ap}_{(-)}(\mathrm{eq}_1 cts)}\text{❦}\Big(\mathrm{ap}_{f\circ\mathrm{rec}_1(D,d,p)}(\mathrm{eq}_1 cts)\Big) && \langle \mathrm{ap}_{a\mapsto \mathrm{transport}^{\mathrm{ap}_{(-)}(\mathrm{eq}_1 cts)}a(\ldots)}(\text{HoTT book, 2.6.2})\rangle \\
&= \mathrm{transport}^{(-)(\mathrm{eq}_1 cts)}\Big(\mathrm{ap}_{\mathrm{ap}}\text{❦}\Big)\Big(\mathrm{ap}_{f\circ\mathrm{rec}_1(D,d,p)}(\mathrm{eq}_1 cts)\Big) && \langle \text{HoTT book, 2.3.10}\rangle \\
&= \mathrm{ap}_{\mathrm{rec}_1(E,e,q)}(\mathrm{eq}_1 cts). && \langle \mathrm{apd}_{(-)(\mathrm{eq}_1 cts)}\Big(\mathrm{ap}_{\mathrm{ap}}\text{❦}\Big)\rangle
\end{aligned}
$$

Finally, for the right side, we observe that the following diagram commutes for any generic path $\mathrm{pair}^{=}pq : (a_1,b_1) =_{\sum_{a:A} Ba} (a_2,b_2)$ constructed using $\mathrm{pair}^{=}$. This is shown by induction on p alone (because, if we choose the obvious proof for tr-fun, then the computation rules will do the rest).

$$
\begin{array}{ccc}
(a_1,b_1) & \xrightarrow{\quad \mathrm{transportconst}_p^{\sum_{a:A} Ba}(a_1,b_1) \quad} & \mathrm{transport}^{-\mapsto \sum_{a:A} Ba}p(a_1,b_1) \\
{\scriptstyle \mathrm{pair}^{=}pq}\ \Big| & & \Big|\ {\scriptstyle \text{tr-fun } (a\mapsto(a,(-)))p(a_1,b_1)} \\
(a_2,b_2) & \xrightarrow{\quad \mathrm{ap}_{(a_2,(-))}q \quad} & \mathrm{ap}_{(a_2,(-))}q
\end{array}
$$

□

As already observed by S. Awodey, J. Frey, and S. Speight [4], naturality alone in not enough to obtain the initial algebra in our new setting. While Theorem 2.17 also extends to wild categories, as we can't prove that $\mathrm{rec}_1(W_1,\mathrm{sup}_1,\mathrm{eq}_1)$ is the identity on $(W_1,\mathrm{sup}_1,\mathrm{eq}_1)$, we cannot regard out construction as the limit of the identity functor to begin with. All that Lemma 3.17 shows, when setting f to $\mathrm{rec}_1(W_1,\mathrm{sup}_1,\mathrm{eq}_1)$, is that such a homomorphism is an idempotent in $\mathcal{S}$– WSusAlg. If what we are after is, instead, the identity, we should try to split it. We take care of it in the next section.

3.4 W-Suspension Algebras and Quasi-Idempotents

To split a $\mathcal{W}$-suspension algebra morphism, it is only reasonable to start from the underlying function on the $\mathcal{W}$-suspension algebra carrier, which is an idempotent in the wild category built on our type universe. We call these *pre-idempotents*.

Definition 3.18. A *pre-idempotent on* X is an endofunction $f : X \to X$ equipped with a homotopy $I : f\circ f \sim f$.

$$\text{PIdem } X :\equiv \sum_{f:X\to X} (f\circ f \sim f)$$

Note that pre-idempotents and all their derived notions are only defined within the wild category build on our type universe. Neither M. Shulman [22], who first introduced them, nor us (need to) extend these notions to other categories. Now then, a splitting operation would result in the following, with $f \sim s\circ r$.

Definition 3.19. A *retraction on* $X : \mathcal{U}$ is a type $A : \mathcal{U}$ equipped with two functions $r : X \to A, s : A \to X$ and a homotopy $r\circ s \sim \mathrm{id}_A$.

$$\text{Retr } X :\equiv \sum_{A:\mathcal{U}} \sum_{r:X\to A} \sum_{s:A\to X} (r\circ s \sim \mathrm{id}_A)$$

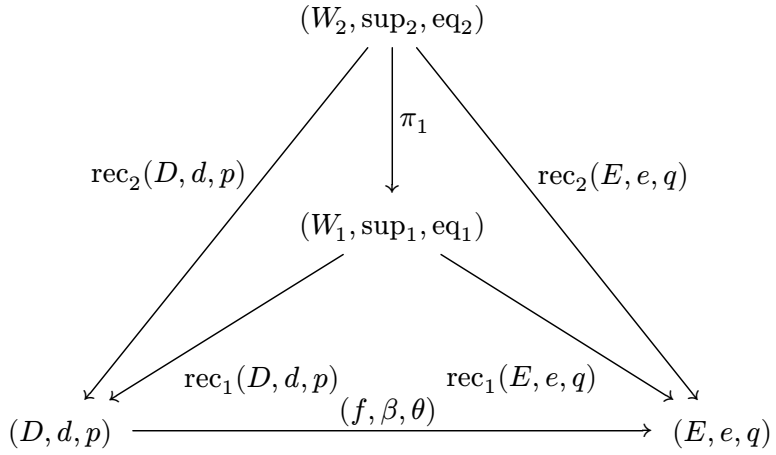
By assuming a natural numbers type, M. Shulman [22] identifies a sufficient condition for a pre-idempotent to split: being a quasi-idempotent.

Definition 3.20. A *quasi-idempotent on* $X : \mathcal{U}$ is a pre-idempotent such that $\mathrm{ap}_f I \sim I\circ f$.

$$\text{QIdem } X :\equiv \sum_{(f,I):\ \text{PIdem } X} \big(\mathrm{ap}_f I \sim I\circ f\big)$$

To obtain a quasi-idempotent, we just need to climb one more step of our refinement hierarchy. By doing so, we don't lose any of the properties proven insofar.

Remark. Via the inclusion map π_1, Lemma 3.17 still holds if we replace the $\mathcal{W}$-suspension algebra morphism on rec_1 with the one on rec_2. In the following diagram, the left and the right triangles commute definitionally, while the bottom one does so by Lemma 3.17.



In particular, the $\mathcal{W}$-suspension algebra morphism given by

$$\mathrm{rec}_2(W_2, \mathrm{sup}_2, \mathrm{eq}_2) \equiv \mathrm{rec}_1(W_2, \mathrm{sup}_2, \mathrm{eq}_2)\circ\pi_1 : W_2 \to W_2$$

is, again, an idempotent in $\mathcal{S}$– WSusAlg. This also entails that the $\mathrm{rec}_2(W_2, \mathrm{sup}_2, \mathrm{eq}_2)$ as a function is a pre-idempotent.

Lemma 3.21. The pre-idempotent on $\mathrm{rec}_2(W_2, \mathrm{sup}_2, \mathrm{eq}_2) : W_2 \to W_2$ identified by the previous Remark extends to a quasi-idempotent.

Proof. We shall abbreviate $(W_2, \mathrm{sup}_2, \mathrm{eq}_2)$ as $\mathbb{W}$. For starters, let's spell out the witness of idempotency given by the previous proof. It is

$$I : \mathrm{rec}_2\mathbb{W} \circ \mathrm{rec}_2\mathbb{W} \sim \mathrm{rec}_2\mathbb{W}$$
$$I :\equiv \lambda(x, i, j) : W_2.i\mathcal{D}$$

where $\mathcal{D} :\equiv (D, d, p)$ is a fibered $\mathcal{W}$-suspension algebra on $(W_0, \mathrm{sup}_0, \mathrm{eq}_0)$. Its algebra structure is the following.

$$D : W_0 \to \mathcal{U}$$
$$D\alpha :\equiv \mathrm{rec}_2\mathbb{W}(\alpha\mathbb{W}) =_{W_2} \alpha\mathbb{W}$$
$$d : \prod_{a:A} \prod_{t:Ba \to W_0} \left(\prod_{b:Ba} D(tb) \right) \to D(\mathrm{sup}_2 at)$$
$$datg :\equiv \mathrm{refl}_{\mathrm{rec}_2\mathbb{W}(\mathrm{sup}_2 at\mathbb{W})} \blacksquare \mathrm{ap}_{\mathrm{sup}_2 a}(\mathrm{funext}\ g)$$

During the proof, we will never need to use our previous construction for p, but recalling its type is at least useful to ensure everything typechecks.

$$p : \prod_{c:C} \prod_{t:B(lc) \to W_0} \prod_{s:B(rc) \to W_0} \prod_{u:\prod_{b:B(lc)} D(tb)} \prod_{v:\prod_{b:B(rc)} D(sb)} \Big(\mathrm{transport}^D(\mathrm{eq}_0 cts)(d(lc)tu) =_{D(\mathrm{sup}_0(rc)s)} d(rc)sv\Big)$$

We are looking for a term

$$J : \mathrm{ap}_{\mathrm{rec}_2\mathbb{W}} \circ I \sim I \circ \mathrm{rec}_2\mathbb{W}.$$

Let $(x, j) : W_2$, with $x : W_1$. We need to show

$$\Big(\mathrm{ap}_{\mathrm{rec}_2\mathbb{W}} \circ I\Big)(x, j) = (I \circ (\mathrm{rec}_2\mathbb{W}))(x, j).$$

Our goal type reduces to

$$\Big(\mathrm{ap}_{\mathrm{rec}_2\mathbb{W}} \circ (-)\mathcal{D} \circ \mathrm{pr}_2 \circ \mathrm{pr}_1\Big)(x, j) = ((-)\mathcal{D} \circ \mathrm{pr}_2 \circ \mathrm{pr}_1 \circ (-)\mathbb{W} \circ \rho_2)(x, j)$$

and then

$$Ex :\equiv \Big(\mathrm{ap}_{\mathrm{rec}_2\mathbb{W}} \circ (-)\mathcal{D} \circ \mathrm{pr}_2\Big)x = ((-)\mathcal{D} \circ \mathrm{pr}_2 \circ \mathrm{pr}_1 \circ (-)\mathbb{W} \circ \mathrm{pr}_1)x.$$

This can be seen as a type family $E : W_1 \to \mathcal{U}$ in the variable $x : W_1$, so j allows us to prove the statement inductively. For the point constructors, let $a : A, t : Ba \to W_1, g : \prod_{b:Ba} E(tb)$. We have:

$$
\begin{aligned}
&\Big(\mathrm{ap}_{\mathrm{rec}_2\mathbb{W}}\circ(-)\mathcal{D}\circ\mathrm{pr}_2\Big)(\mathrm{sup}_1 at)\\
&\equiv \mathrm{ap}_{\mathrm{rec}_2\mathbb{W}}(da(\mathrm{pr}_1\circ t)((-)\mathcal{D}\circ\mathrm{pr}_2\circ t))\\
&\equiv \mathrm{ap}_{\mathrm{rec}_2\mathbb{W}}\Big(\mathrm{refl}\circ\mathrm{ap}_{\mathrm{sup}_2 a}(\mathrm{funext}\ ((-)\mathcal{D}\circ\mathrm{pr}_2\circ t))\Big)\\
&= \mathrm{ap}_{\mathrm{rec}_2\mathbb{W}}\Big(\mathrm{ap}_{\mathrm{sup}_2 a}(\mathrm{funext}\ ((-)\mathcal{D}\circ\mathrm{pr}_2\circ t))\Big) && \langle \mathrm{ap}_{\mathrm{ap}_{\mathrm{rec}_2\mathbb{W}}}\Big(\mathrm{reflL}\ \Big(\mathrm{ap}_{\mathrm{sup}_2 a}(\mathrm{funext}\ ((-)\mathcal{D}\circ\mathrm{pr}_2\circ t))\Big)\Big)\rangle\\
&= \mathrm{ap}_{\mathrm{rec}_2\mathbb{W}\circ\mathrm{sup}_2 a}(\mathrm{funext}\ ((-)\mathcal{D}\circ\mathrm{pr}_2\circ t)) && \langle \text{ap-comp}\ (\mathrm{rec}_2\mathbb{W})(\mathrm{sup}_2 a)(\mathrm{funext}\ ((-)\mathcal{D}\circ\mathrm{pr}_2\circ t))\rangle\\
&\equiv \mathrm{ap}_{\mathrm{sup}_2 a(\mathrm{rec}_2\mathbb{W}\circ(-))}(\mathrm{funext}\ ((-)\mathcal{D}\circ\mathrm{pr}_2\circ t))\\
&= \mathrm{ap}_{\mathrm{sup}_2 a}\Big(\mathrm{ap}_{\mathrm{rec}_2\mathbb{W}\circ(-)}(\mathrm{funext}\ ((-)\mathcal{D}\circ\mathrm{pr}_2\circ t))\Big) && \langle (\text{ap-comp}\ (\mathrm{sup}_2 a)(\mathrm{rec}_2\mathbb{W}\circ(-))(\mathrm{funext}\ ((-)\mathcal{D}\circ\mathrm{pr}_2\circ t)))^{-1}\rangle\\
&= \mathrm{ap}_{\mathrm{sup}_2 a}(\mathrm{funext}\ (\mathrm{rec}_2\mathbb{W}\circ(-)\mathcal{D}\circ\mathrm{pr}_2\circ t)) && \langle \mathrm{ap}_{\mathrm{ap}_{\mathrm{sup}_2 a}}(\text{X. Ripoll Echeveste [21], lemma 3.12})\rangle\\
&= \mathrm{ap}_{\mathrm{sup}_2 a}(\mathrm{funext}\ ((-)\mathcal{D}\circ\mathrm{pr}_2\circ\mathrm{pr}_1\circ(-)\mathbb{W}\circ\mathrm{pr}_1\circ t)) && \langle \mathrm{ap}_{\mathrm{ap}_{\mathrm{sup}_2 a}\circ\ \mathrm{funext}}(\mathrm{funext}\ g)\rangle\\
&= \mathrm{refl}\blacksquare\mathrm{ap}_{\mathrm{sup}_2 a}(\mathrm{funext}\ ((-)\mathcal{D}\circ\mathrm{pr}_2\circ\mathrm{pr}_1\circ(-)\mathbb{W}\circ\mathrm{pr}_1\circ t)) && \langle\Big(\mathrm{reflL}\ \Big(\mathrm{ap}_{\mathrm{sup}_2 a}(\mathrm{funext}\ ((-)\mathcal{D}\circ\mathrm{pr}_2\circ\mathrm{pr}_1\circ(-)\mathbb{W}\circ\mathrm{pr}_1\circ t))\Big)\Big)^{-1}\rangle\\
&\equiv da(\rho_2\circ(-)\mathbb{W}\circ\mathrm{pr}_1\circ t)((-)\mathcal{D}\circ\mathrm{pr}_2\circ\mathrm{pr}_1\circ(-)\mathbb{W}\circ\mathrm{pr}_1\circ t)\\
&\equiv ((-)\mathcal{D}\circ\mathrm{pr}_2)(\mathrm{sup}_1 a(\mathrm{pr}_1\circ(-)\mathbb{W}\circ\mathrm{pr}_1\circ t))\\
&\equiv ((-)\mathcal{D}\circ\mathrm{pr}_2\circ\mathrm{pr}_1\circ(-)\mathbb{W}\circ\mathrm{pr}_1)(\mathrm{sup}_1 at)
\end{aligned}
$$

For the path constructors, let $c : C, t : B(lc) \to W_1, s : B(rc) \to W_1, u : \prod_{b:B(lc)} E(tb), v : \prod_{b:B(rc)} E(sb)$. If we name the proof term for out induction step on point constructors γ, we have to show

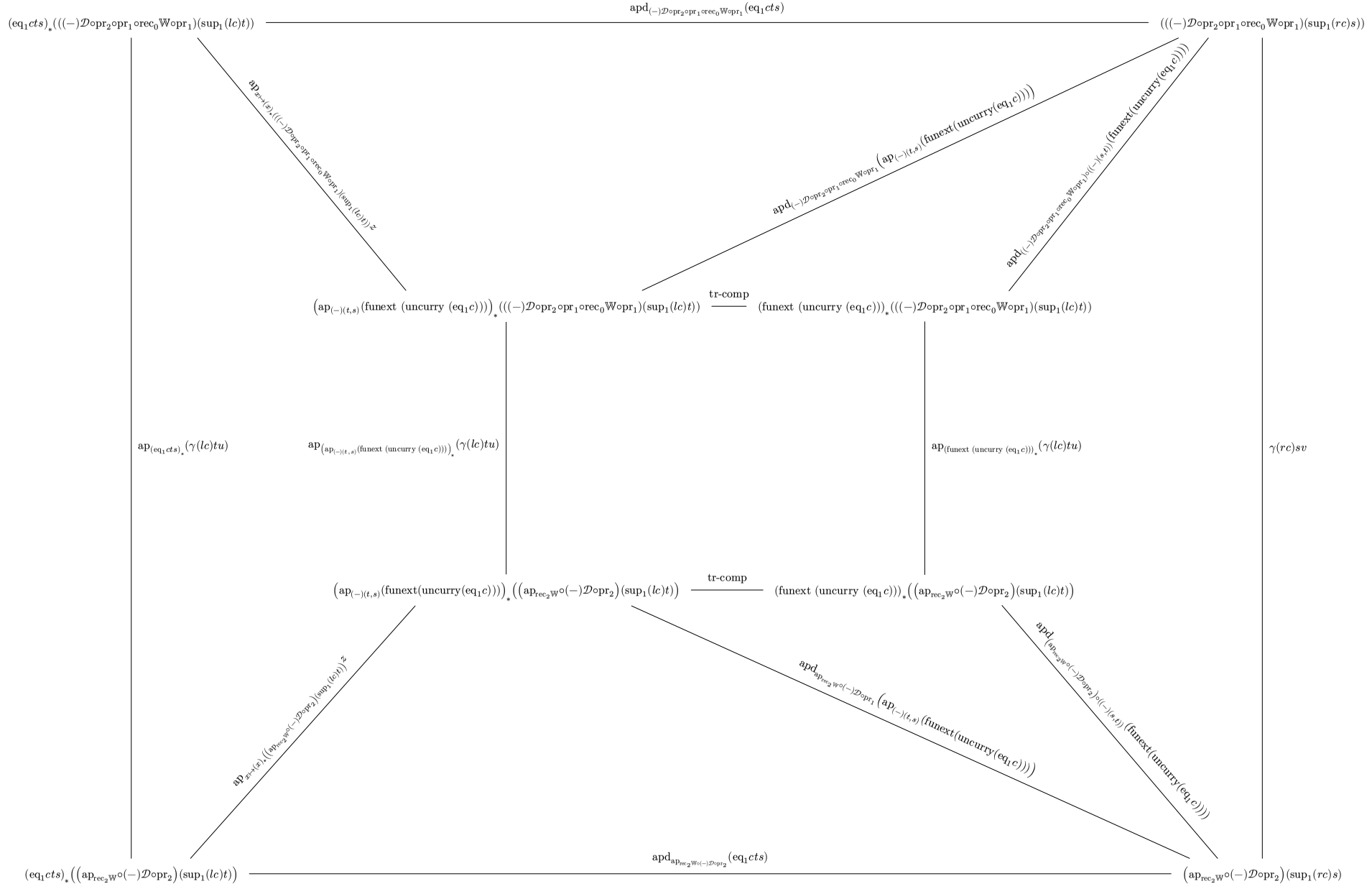
$$\mathrm{transport}^D(\mathrm{eq}_1 cts)(\gamma(lc)tu) =_{D(rc)s} \gamma(rc)sv.$$

The trick is to rewrite the path we are transporting along, $\mathrm{eq}_1 cts$, in a different form:

$$z : \mathrm{eq}_1 cts =_{\mathrm{sup}_1(lc)t =_{W_1} \mathrm{sup}_1(rc)s} \mathrm{ap}_{(-)(t,s)}(\mathrm{funext}\ (\mathrm{uncurry}\ (\mathrm{eq}_1 c))$$

$$\mathrm{eq}_1 cts \equiv (\mathrm{uncurry}\ \mathrm{eq}_1 c)(t,s) \overset{(\text{ap-funext}\ (t,s)(\mathrm{uncurry}\ (\mathrm{eq}_1 c)))^{-1}}{\underline{\qquad\qquad}} \mathrm{ap}_{(-)(t,s)}(\mathrm{funext}\ (\mathrm{uncurry}\ (\mathrm{eq}_1 c)))$$

The next commutative diagram proves our goal. The central square commutes because of the dependent action of tr-comp on the path $\gamma(lc)tu$. The right trapezoid because its two bases can be seen as the application of the same function to $\mathrm{sup}_1(lc)\circ\mathrm{pr}_1)$ and $\mathrm{sup}_1(rc)\circ\mathrm{pr}_2$ respectively, ando so we can use the dependent action of such a function on path funext (uncurry $(\mathrm{eq}_1 c)$). The left trapezoid commutes because of the dependent action of $\mathrm{ap}_{(-)_*}(\gamma a(lc)u)$ on z. Both triangles on the left (top and bottom) can be seen to commute after generalising the statement until we can perform path induction on the variable generalising z. Similarly, both triangles on the right (top and bottom) also commute, but this time we need to generalise until we can make path induction on the variable generalsing (funext (uncurry$(\mathrm{eq}_1 c)$)).


$(\mathrm{eq}_1 cts)_*(((-)\mathcal{D}\circ \mathrm{pr}_2\circ \mathrm{pr}_1\circ \mathrm{rec}_0\mathbb{W}\circ \mathrm{pr}_1)(\sup_1(lc)t))$
$\mathrm{apd}_{(-)\mathcal{D}\circ \mathrm{pr}_2\circ \mathrm{pr}_1\circ \mathrm{rec}_0\mathbb{W}\circ \mathrm{pr}_1}(\mathrm{eq}_1 cts)$
$(((-)\mathcal{D}\circ \mathrm{pr}_2\circ \mathrm{pr}_1\circ \mathrm{rec}_0\mathbb{W}\circ \mathrm{pr}_1)(\sup_1(rc)s))$
$\mathrm{ap}_{x\mapsto(x)_*(((-)\mathcal{D}\circ \mathrm{pr}_2\circ \mathrm{pr}_1\circ \mathrm{rec}_0\mathbb{W}\circ \mathrm{pr}_1)(\sup_1(lc)t))z}$
$\mathrm{apd}_{(-)\mathcal{D}\circ \mathrm{pr}_2\circ \mathrm{pr}_1\circ \mathrm{rec}_0\mathbb{W}\circ \mathrm{pr}_1}\left(\mathrm{ap}_{(-)(t,s)}(\mathrm{funext}(\mathrm{uncurry}(\mathrm{eq}_1 c)))\right)$
$\mathrm{apd}_{(-)\mathcal{D}\circ \mathrm{pr}_2\circ \mathrm{pr}_1\circ \mathrm{rec}_0\mathbb{W}\circ \mathrm{pr}_1\circ((-)(s,t))}(\mathrm{funext}(\mathrm{uncurry}(\mathrm{eq}_1 c)))$
$\left(\mathrm{ap}_{(-)(t,s)}(\mathrm{funext}\ (\mathrm{uncurry}\ (\mathrm{eq}_1 c)))\right)_*(((-)\mathcal{D}\circ \mathrm{pr}_2\circ \mathrm{pr}_1\circ \mathrm{rec}_0\mathbb{W}\circ \mathrm{pr}_1)(\sup_1(lc)t))$
tr-comp
$(\mathrm{funext}\ (\mathrm{uncurry}\ (\mathrm{eq}_1 c)))_*(((-)\mathcal{D}\circ \mathrm{pr}_2\circ \mathrm{pr}_1\circ \mathrm{rec}_0\mathbb{W}\circ \mathrm{pr}_1)(\sup_1(lc)t))$
$\mathrm{ap}_{(\mathrm{eq}_1 cts)_*}(\gamma(lc)tu)$
$\mathrm{ap}_{\left(\mathrm{ap}_{(-)(t,s)}(\mathrm{funext}\ (\mathrm{uncurry}\ (\mathrm{eq}_1 c)))\right)_*}(\gamma(lc)tu)$
$\mathrm{ap}_{(\mathrm{funext}\ (\mathrm{uncurry}\ (\mathrm{eq}_1 c)))_*}(\gamma(lc)tu)$
$\gamma(rc)sv$
$\left(\mathrm{ap}_{(-)(t,s)}(\mathrm{funext}(\mathrm{uncurry}(\mathrm{eq}_1 c)))\right)_*\left(\left(\mathrm{ap}_{\mathrm{rec}_2\mathbb{W}\circ(-)\mathcal{D}\circ \mathrm{pr}_2}\right)(\sup_1(lc)t)\right)$
tr-comp
$(\mathrm{funext}\ (\mathrm{uncurry}\ (\mathrm{eq}_1 c)))_*\left(\left(\mathrm{ap}_{\mathrm{rec}_2\mathbb{W}\circ(-)\mathcal{D}\circ \mathrm{pr}_2}\right)(\sup_1(lc)t)\right)$
$\mathrm{ap}_{z\mapsto(x)_*\left(\left(\mathrm{ap}_{\mathrm{rec}_2\mathbb{W}\circ(-)\mathcal{D}\circ \mathrm{pr}_2}\right)(\sup_1(lc)t)\right)z}$
$\mathrm{apd}_{\mathrm{ap}_{\mathrm{rec}_2\mathbb{W}\circ(-)\mathcal{D}\circ \mathrm{pr}_1}\left(\mathrm{ap}_{(-)(t,s)}(\mathrm{funext}(\mathrm{uncurry}(\mathrm{eq}_1 c)))\right)}$
$\mathrm{apd}_{\left(\mathrm{ap}_{\mathrm{rec}_2\mathbb{W}\circ(-)\mathcal{D}\circ \mathrm{pr}_2}\right)\circ((-)(s,t))}(\mathrm{funext}(\mathrm{uncurry}(\mathrm{eq}_1 c)))$
$(\mathrm{eq}_1 cts)_*\left(\left(\mathrm{ap}_{\mathrm{rec}_2\mathbb{W}\circ(-)\mathcal{D}\circ \mathrm{pr}_2}\right)(\sup_1(lc)t)\right)$
$\mathrm{apd}_{\mathrm{ap}_{\mathrm{rec}_2\mathbb{W}\circ(-)\mathcal{D}\circ \mathrm{pr}_2}}(\mathrm{eq}_1 cts)$
$\left(\mathrm{ap}_{\mathrm{rec}_2\mathbb{W}\circ(-)\mathcal{D}\circ \mathrm{pr}_2}\right)(\sup_1(rc)s)$

This concludes the proof. □

While having a quasi-idempotent in the wild category of types and functions gives us a splitting for free in such a wild category, we still need to reconstruct the splitting in $\mathcal{S}$– WSusAlg.

Proposition 3.22. The idempotent in $\mathcal{S}$– WSusAlg given by the previous Remark splits.

Proof. We sometimes work with propositional equalities between functions, as opposed to homotopies, for convenience. Because we are already assuming function extensionality, this is not a problem.

We set $f :\equiv \mathrm{rec}_2(W_2, \mathrm{sup}_2, \mathrm{eq}_2)$, and $(D, d, p) :\equiv (W_2, \mathrm{sup}_2, \mathrm{eq}_2)$. We refer to its witness of pre-idempotency as $I : f \circ f = f$. Theorem 5.3 of M. Shulman [22] gives a splitting of our quasi-idempotent on (f, I). We use the variables $E : \mathcal{U}, \mathit{r} : D \to E, \mathit{s} : E \to D, K : \mathit{r} \circ \mathit{s} = 1_E$ for the splitting, and name $H : \mathit{s} \circ \mathit{r} = f$ its "reconstruction".

The proof takes place in two stages. First, we lift the splitting to the category of regular $\mathcal{W}$-algebras for signature (A, B). Only then we move to the lifting to $\mathcal{S}$– WSusAlg.

First, we show that we can give E the structure of a $\mathcal{W}$-algebra. As a point constructor map, we choose

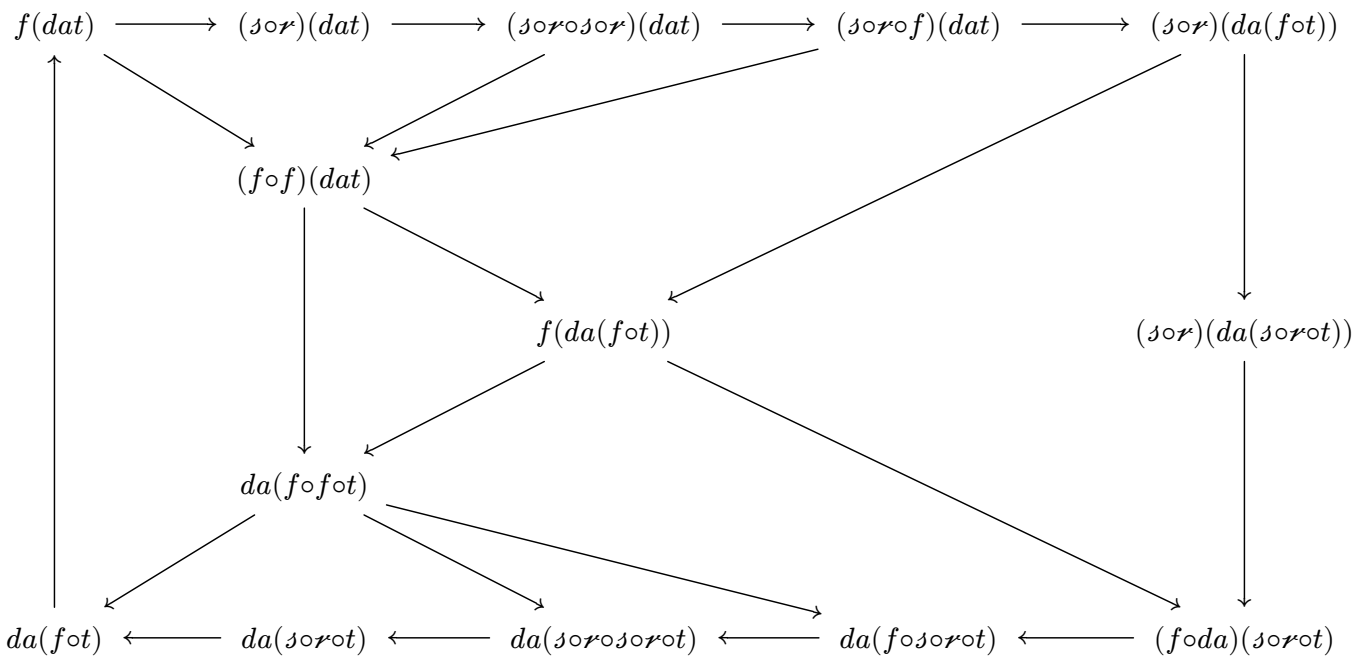
$$e : \prod_{a:A} (Ba \to E) \to E$$
$$ea :\equiv \mathit{r} \circ da(\mathit{s} \circ (-)).$$

Similarly, r and s can be given the structure of algebra moprhisms. The are extended to (r, ρ) and (s, σ) respectively.

$$\begin{aligned}
\rho a &: \mathit{r} \circ da \sim ea(\rho \circ (-)) \\
\rho at &:\equiv \\
\mathit{r}(dat) &= (\mathit{r} \circ \mathit{s} \circ \mathit{r})(dat) && \langle \mathrm{ap}_{((-)\circ \mathit{r})(dat)} K^{-1} \rangle \\
&= (\mathit{r} \circ f)(dat) && \langle \mathrm{ap}_{(\mathit{r} \circ (-)) \circ (dat)} H \rangle \\
&= (\mathit{r} \circ da)(f \circ t) && \langle \mathrm{ap}_{\mathit{r}}(\beta at) \rangle \\
&= (\mathit{r} \circ da)(\mathit{s} \circ \mathit{r} \circ t) && \langle \mathrm{ap}_{(\mathit{r} \circ da)((-)\circ t)} H^{-1} \rangle \\
\sigma a &: \mathit{s} \circ ea \sim da(\mathit{s} \circ (-)) \\
\sigma at &:\equiv \\
(\mathit{s} \circ \mathit{r} \circ da)(\mathit{s} \circ t) &= (f \circ da)(\mathit{s} \circ t) && \langle \mathrm{ap}_{((-)\circ da)(\mathit{s} \circ t)} H \rangle \\
&= da(f \circ \mathit{s} \circ t) && \langle \beta a(\mathit{s} \circ t) \rangle \\
&= da(\mathit{s} \circ \mathit{r} \circ \mathit{s} \circ t) && \langle \mathrm{ap}_{da((-)\circ \mathit{s} \circ t)} H^{-1} \rangle \\
&= da(\mathit{s} \circ t) && \langle \mathrm{ap}_{da(\mathit{s} \circ (-) \circ t)} K \rangle
\end{aligned}$$

We now use $\mathrm{pair}^{=}$ to lift H and K. We need $H' : H_*(\sigma \circ \rho) = \beta$ and $K' : K_*(\rho \circ \sigma) = \lambda at.\mathrm{refl}_{eat}$. For both, we use function extensionality to fix $a : A$ and $t : Ba \to D$ (or $t : Ba \to E$) and check that the following two diagrams commute. As per usual, theorem 2.11.3 from the HoTT book helps us get rid of the transport operator. The higher order paths in the diagrams are not labelled, as they

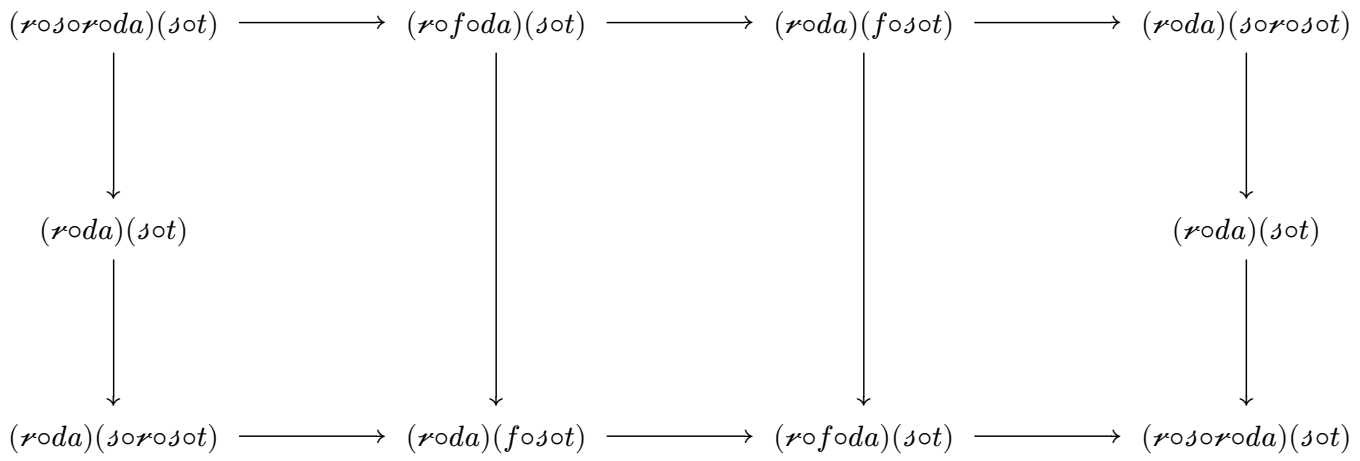
coincide with the definitions of ρ and σ that we have just given, plus the paths we got in exchange for eliminating transport.



The only polygons which do not commute due to simple path algebra properties are:

- the left trapezoid, which commutes due to the algebra morphism on f being an idempotent;
- the center triangle, which is just our definition of morphism composition;
- the triangles at the top left and bottom left. These commute since $H_*\left(\mathrm{ap}_{\jmath\circ(-)\circ\jmath} K\right) = I$, as stated in theorem 7.1 of M. Shulman [22].

We now proceed to the diagram for the second equality.



The right square is the same as the left one, but rotated by 180 degrees. Observe that the proof that f is an algebra morphism is, for our particular choice of $f \equiv \mathrm{rec}_2(W_2, \mathrm{sup}_2, \mathrm{eq}_2)$, pointwise defined using path reflexivity. The center square indeed definitionally degenerates into a point (for a generic algebra morphism, if would still be propositionally equal to refl, as the symmetric paths

would cancel each other out). The left (right) square can be made to commute by being rephrased as a single transport of refl (not along refl) using tr-comp and tr-trans.

This concludes the lifting to regular $\mathcal{W}$-algebras. Now, as a path constructor map for (E, e), we have

$$q : \prod_{c:C} \prod_{t:B(lc)\to E} \prod_{s:B(rc)\to E} (e(lc)t =_E e(rc)s)$$

$$qcts :\equiv \mathrm{transport}^{x\mapsto(e(lc)(x\circ t)=_E e(rc)(x\circ s))} K(\rho(lc)(\jmath\circ t) \blacksquare \mathrm{ap}_r(pc(\jmath\circ t)(\jmath\circ s)) \blacksquare \rho(rc)(\jmath\circ s))$$

The extensions of (s, σ) and (r, ρ) to $\mathcal{W}$-suspension algebra morphisms is quite long an not particularly interesting (chains of 8 and 10 propositional equalities respectively). The path algebra toolkit we have been using so far is adequate for the job. What is important to remember is that the proof that f is a $\mathcal{W}$-suspension algebra morphism is, of course, needed, and that $\mathrm{pair}^{=}HH'$ and $\mathrm{pair}^{=}KK'$ must also be extended accordingly. □

The previous lemma does not lift an arbitrary splitting for a generic quasi-idempotent. It is leveraging our particular construction for $\mathrm{rec}_2(W_2, \mathrm{sup}_2, \mathrm{eq}_2)$.

3.5 Initiality

We are now able to prove initiality of the $\mathcal{W}$-suspension algebra $\left(W^{\mathcal{S}}, \mathrm{sup}^{\mathcal{S}}, \mathrm{eq}^{\mathcal{S}}\right)$ obtained during the previous splitting in the wild category $\mathcal{S}$– WSusAlg.

Lemma 3.23. Theorem 2.17 can be extended (also in wild categories) with this third, equivalent condition: "I is the vertex of a cone for the identity functor and the projection from I to I itself is 1_I".

Proof. It is obvious to see that this property is implied by initiality. To see the converse, consider a generic projection $\mu_X : I \to X$. Because I is the vertex of a cone on the identity functor, any other map $f : I \to X$ satisfies

$$f = f\circ 1_I = f\circ\mu_I = \mu_X.$$

□

The $\mathcal{W}$-suspension algebra that is the result of our splitting certainly enjoys these two conditions.

Theorem 3.24. There is an initial object $\left(W^{\mathcal{S}}, \mathrm{sup}^{\mathcal{S}}, \mathrm{eq}^{\mathcal{S}}\right)$ in $\mathcal{S}$– WSusAlg.

Proof. We use Lemma 3.23. As a candidate for the initial object, we choose $I :\equiv \left(W^{\mathcal{S}}, \mathrm{sup}^{\mathcal{S}}, \mathrm{eq}^{\mathcal{S}}\right)$, which we obtained via the splitting in Proposition 3.22. Fixed a $\mathcal{W}$-suspension algebra $\mathcal{D}$, we have projection $\pi_{\mathcal{D}}$ from I to $\mathcal{D}$ which is obtained by composing the morphism extending $\mathrm{rec}_2\mathcal{D}$ after $\mathfrak{S}$, the section obtained from the splitting. This choice ensures that we get a cone on the identity functor, since $f\circ\mathrm{rec}_2\mathcal{D} = \mathrm{rec}_2\mathcal{E}$ holds from naturality of $(W_2, \mathrm{sup}_2, \mathrm{eq}_2)$, and hence $f\circ\pi_{\mathcal{D}} \equiv f\circ\mathrm{rec}_2\mathcal{D}\circ\mathfrak{S} = \mathrm{rec}_2\mathcal{E}\circ\mathfrak{S} = \pi_{\mathcal{E}}$. So all that is left to show is that $\pi_I =\equiv \mathrm{rec}_2 I\circ\mathfrak{S} = 1_I$. Let $\mathfrak{R}$ be the retraction obtained from the splitting. Because $\mathfrak{R}\circ\mathfrak{S} = 1_I$, it is enough to show that $\mathfrak{R} = \mathrm{rec}_2 I$. Indeed, we have

$$
\begin{aligned}
\mathfrak{R} &= 1_I \circ \mathfrak{R} \\
&= \mathfrak{R} \circ \mathfrak{S} \circ \mathfrak{R} \\
&= \mathfrak{R} \circ \mathrm{rec}_2(W_2, \mathrm{sup}_2, \mathrm{eq}_2) \\
&= \mathrm{rec}_2 I. && \langle \text{naturality} \rangle
\end{aligned}
$$

□

Corollary 3.25. By Theorem 3.10, the $\mathcal{W}$-suspension algebra $(W^2, \mathrm{sup}^2, \mathrm{eq}^2)$ satisfies the induction principle for $\mathcal{W}$-suspension algebras over $\mathcal{S}$.

Similarly to what we have done in the previous chapter, the encoding of the $\mathcal{W}$-suspension is given by $W^{\mathcal{S}}$. The encoding of the point constructor with label $a : A$ is given by $\mathrm{sup}^{\mathcal{S}} a$. The encoding of the path constructor with label $c : C$ is given by $\mathrm{eq}^{\mathcal{S}} c$.

Chapter 4

Formalisation

Both the code used to typeset this thesis and the formalisation itself are available at the following repository.

`https://codeberg.org/foxy/impredicative-encodings-of-hits`

The offline reader may instead consult the compressed archive embedded within this PDF document. While the markup code for the thesis consists of a Typst project released under Creative Commons Attribution-NoDerivatives 4.0 International, the formalisation is distributed as a self-contained Agda library under the GNU Affero General Public License. This setup makes it fairly easy to build upon our formalisation.

The project is built against Agda v2.8.0, the latest stable release at the time of writing. The following sections touch upon more technical design choices.

4.1 Project Structure

The following tree omits markup code and assets for the thesis, as well as licences, `git`-related files, CI/CD pipelines, and `README.md`.

```
/
├─ impredicative-encodings-of-hits.agda-lib (library file)
└─ src/
   ├─ ImprHIT.agda
   ├─ Background.agda
   ├─ Background/
   │  ├─ Weide.agda (Section 1.1.1)
   │  ├─ WSuspensions.agda (Section 1.1.2)
   │  ├─ Examples.agda (Section 1.1.3)
   │  └─ Impredicativity.agda (Section 1.3)
   ├─ SetTruncated.agda
   ├─ SetTruncated/
   │  ├─ LawfulSetAlgebra.agda (Section 2.1)
   │  ├─ Limits.agda (Section 2.2)
   │  ├─ Initial.agda (Section 2.3)
   │  └─ Rules.agda (Section 2.4)
   └─ General.agda
```

4.2 Cubical Mode

As already stated, univalence is not assumed anywhere in our work. All the formalised results already hold in Martin-Löf's intensional type theory enriched with function extensionality. However, because homotopy levels and their theory historically originate in homotopy type theory, vanilla, non-univalent Agda does not feature those in its standard library. Instead, these are part of Agda's Cubical library [3]. Cubical mode (`--cubical`) is how Agda approaches homotopy type theory and univalent foundations, as described by A. Vezzosi, A. Mörtberg, and A. Abel [28]. With the goal of not reinventing the wheel in mind, our set-truncated encodings are therefore formalised in Cubical Agda. Of course, the added definitional equalities also make the formalisation work itself easier. At the time of writing, the `agda/cubical` library has yet to see its first stable release, so expect the build to fail if you upgrade it without adapting the codebase. Much like this library, we also make use of the `--no-import-sorts` flag. This ensures we consistently use the `Type` nomenclature introduced by `agda/cubical` for type universes, rather than the misleading `Set`. Because the library makes project-wide use of `--guardedness`, we have to use such an option, too. The last flag our project shares with the standard library is `-WnoUnsupportedIndexedMatch`. This ignores warnings about pattern-matching features that are safe, but whose computational behaviour in presence of transports has not been implemented by Cubical Agda yet.

From a stylistic point of view, we try to avoid any explicit mention of interval variables in our proofs. Instead, we work at a higher abstraction level by making use of well-known lemmas already defined in `agda/cubical`, such as congruence, symmetry, and function extensionality. This makes our work fully understandable to people with experience in homotopy type theory, but not in cubical type theory.

In a way, because the literature on dependent type theory is lacking rigorous conservativity results between the different type theories that extend our basic system, rewriting our proof in this specific flavour of Cubical Type Theory could even be seen as its own result.

4.3 Extensibility

A possible future formalisation of the non-truncated encodings would not require any theory about homotopy levels. Hence, it might be sensible not to use Cubical Agda for it. Our setup takes this into account by marking the project as `--cubical-compatible`, rather than `--cubical`, and using `--cubical` only in the modules in which it is actually required. The necessary scaffolding for the future work, as well as a definition of $\mathcal{W}$-suspension signature inspired by the `Containers` found in the Agda standard library [2], are already in place.

4.4 Impredicativity

Agda's bottom universe is predicative, while our work necessitates impredicative products. We could, of course, recreate our variant of homotopy type theory within Agda's type system, using the latter as our meta-theory, and then working in our object theory. This, however, would prove to be too cumbersome. Instead, we just axiomatise impredicative products as their own, separate types next to Agda's predicative ones. That is to say, we postulate the types of a type former `Π`, a term constructor `λ`, an eliminator `ev`.

```
11  postulate
12    Π : (A : Type ℓ) (B : A → Type₀) → Type₀ -- formation
13    Λ : (∀ x → B x) → Π A B                  -- introduction
```

Listing 1: `Background.Impredicativity.agda`, lines 11-14 ⚙

The β-rule `Π-β` and the η-rule `Π-η` look like the following:

```
16  postulate
17    Π-β : (f : ∀ x → B x) → ev (Λ f) ≡ f
18    Π-η : (f : Π A B) → Λ (ev f) ≡ f
19
20  {-# BUILTIN REWRITE _≡_ #-}
```

Listing 2: `Background.Impredicativity.agda`, lines 16-21 ⚙

The last two lines state that Agda's evaluation relation should be extended with these two new computation rules. The code snippet for non-Cubical Agda is equivalent. These postulates, next to the use of `REWRITE`, are the only reason our project cannot qualify as `--safe`. People not familiar with Agda syntax might be surprised at $\equiv$ being used as a binary dependent type. Confusingly, this symbol is used by Agda to denote *propositional* equality. So what we are actually doing here is writing down propositional versions of our rules inside the system first, and only later turning them into rewrite rules.

Chapter 5

Conclusions

We have extended the set-truncated impredicative encodings given by S. Awodey, J. Frey, and S. Speight [4] from $\mathcal{W}$-types to Van der Weide HITs, formalising our work in Cubical Agda. We have extended the impredicative encodings given by X. Ripoll Echeveste [21] from $\mathcal{W}$-types to $\mathcal{W}$-suspensions.

5.1 Related Work

The seminal paper by S. Awodey, J. Frey, and S. Speight [4] already describes the encoding of the circle and of 1-truncations, without generalising to any class of higher inductive types. The encodings for the regular inductive types are adapted to quotients and M-types by S. Bronsveld, H. Geuvers, and N. van der Weide [8], who base their work on S. Bronsveld [9]. While their paper assumes uniqueness of identity proofs, one could in principle port their results to a system without UIP, by working within the impredicative subuniverse of set-truncated types.

If we move to encodings of HITs that reduce them to simpler ones, rather than employing impredicativity, the literature becomes much richer. F. van Doorn [11] and N. Kraus [15] use non-recursive higher inductive types to construct propositional truncation. E. Rijke [19] starts from the join construction to encode n-truncations, the Rezk completion, and set quotients. A. Kaposi, A. Kovács, and T. Altenkirch [14] show that all finitary quotient inductive-inductive types can be reconstructed starting from one particular member of such a class and uniqueness of identity proofs. N. van der Weide and H. Geuvers [29], working with the same Van der Weide higher inductive types we also used, reduce all set-truncated HITs to quotients and propositional truncations. Finally, N. van der Weide [30], with a more general class that also allows for homotopy constructors, encodes all 1-truncated HITs from the groupoid quotient. All these methods take place entirely within the chosen type theory, whatever that may be.

5.2 Limitations

While our work does not necessitate any additional assumptions when compared to its counterparts for $\mathcal{W}$-types, it certainly suffers from the same limitation: the obtained encodings do not allow for large elimination, i.e. elimination into higher universes.

5.3 Future Work

The most immediate path of work is adapting our non-truncated encodings to classes of higher inductive types that are more and more general. A. Kaposi and A. Kovács [13] list several candidates. However, not much work has been done to relate different classes of higher inductive types in terms of expressivity. Secondly, reworking our proof with an exposition informed by 2-category theory

would be able to give more insight on *why* two layers of inductivity suffice. Finally, our setup for impredicativity in Agda can easily be reused as-is for our non-truncated impredicative encodings of $\mathcal{W}$-suspensions.

References

[1] Agda Community. 2024. Agda. Retrieved from https://wiki.portal.chalmers.se/agda

[2] Agda Community. 2024. Agda Standard Library. Retrieved from https://github.com/agda/agda-stdlib

[3] Agda Community. 2025. Cubical Agda Library. Retrieved from https://github.com/agda/cubical

[4] Steve Awodey, Jonas Frey, and Sam Speight. 2018. Impredicative Encodings of (Higher) Inductive Types. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '18)*, July 2018. ACM, 76–85. https://doi.org/10.1145/3209108.3209130

[5] Steve Awodey, Nicola Gambino, and Kristina Sojakova. 2012. Inductive Types in Homotopy Type Theory. In *Logic in Computer Science, Symposium on*, June 2012. IEEE Computer Society, 95–104. https://doi.org/10.1109/LICS.2012.21

[6] Steve Awodey, Nicola Gambino, and Kristina Sojakova. 2017. Homotopy-Initial Algebras in Type Theory. *J. ACM* 63, 6 (January 2017). https://doi.org/10.1145/3006383

[7] Henning Basold, Herman Geuvers, and Niels van der Weide. 2017. Higher Inductive Types in Programming. *23*, 63–88. https://doi.org/10.3217/jucs-023-01-0063

[8] Steven Bronsveld, Herman Geuvers, and Niels van der Weide. 2025. Impredicative Encodings of Inductive and Coinductive Types. 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. https://doi.org/10.4230/LIPICS.FSCD.2025.11

[9] Steven Bronsveld. 2024. Impredicative Encodings of Inductive and Coinductive Types. Master's thesis. https://doi.org/10.48550/arXiv.2505.13495

[10] Paolo Capriotti and Nicolai Kraus. 2017. Univalent higher categories via complete Semi-Segal types. *Proceedings of the ACM on Programming Languages* 2, POPL (December 2017), 1–29. https://doi.org/10.1145/3158132

[11] Floris van Doorn. 2016. Constructing the propositional truncation using non-recursive HITs. In *Proceedings of the 5th ACM SIGPLAN Conference on Certified Programs and Proofs (CPP 2016)*, January 2016. ACM, 122–129. https://doi.org/10.1145/2854065.2854076

[12] Dan Frumin, Herman Geuvers, Léon Gondelman, and Niels van der Weide. 2018. Finite sets in homotopy type theory. In *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP 2018)*, 2018. Association for Computing Machinery, 201–214. https://doi.org/10.1145/3167085

[13] Ambrus Kaposi and András Kovács. 2020. Signatures and Induction Principles for Higher Inductive-Inductive Types. *Logical Methods in Computer Science* (February 2020). https://doi.org/10.23638/lmcs-16(1:10)2020

[14] Ambrus Kaposi, András Kovács, and Thorsten Altenkirch. 2019. Constructing quotient inductive-inductive types. *Proceedings of the ACM on Programming Languages* 3, POPL (January 2019), 1–24. https://doi.org/10.1145/3290315

[15] Nicolai Kraus. 2016. Constructions with Non-Recursive Higher Inductive Types. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '16)*, July 2016. ACM, 595–604. https://doi.org/10.1145/2933575.2933586

[16] Tom Leinster. 2014. *Basic Category Theory*. Cambridge University Press. https://doi.org/10.1017/CBO9781107360068

[17] Per Martin-Löf. 1975. An Intuitionistic Theory of Types: Predicative Part. In *Logic Colloquium '73, Proceedings of the Logic Colloquium*. Elsevier, 73–118. https://doi.org/10.1016/s0049-237x(08)71945-1

[18] Per Martin-Löf. 1982. Constructive Mathematics and Computer Programming. In *Logic, Methodology and Philosophy of Science VI, Proceedings of the Sixth International Congress of Logic, Methodology and Philosophy of Science*. Elsevier, 153–175. https://doi.org/10.1016/s0049-237x(09)70189-2

[19] Egbert Rijke. 2017. The join construction. https://doi.org/10.48550/ARXIV.1701.07538

[20] Egbert Rijke. 2025. *Introduction to Homotopy Type Theory*. Cambridge University Press. Retrieved from https://www.cambridge.org/core/books/introduction-to-homotopy-type-theory/0DD31EC06C80797A50ACE807251E80B6

[21] Xavier Ripoll Echeveste. 2023. Alternative Impredicative Encodings of Inductive Types. Master's thesis. Retrieved from https://eprints.illc.uva.nl/id/eprint/2272

[22] Michael Shulman. 2017. Idempotents in intensional type theory. *Logical Methods in Computer Science* (April 2017). https://doi.org/10.2168/lmcs-12(3:9)2016

[23] Michael Shulman. 2018. Impredicative Encodings, Part 3. *Homotopy Type Theory*. Retrieved from https://homotopytypetheory.org/2018/11/26/impredicative-encodings-part-3

[24] Kristina Sojakova. 2015. Higher Inductive Types as Homotopy-Initial Algebras. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '15)*, January 2015. ACM, 31–42. https://doi.org/10.1145/2676726.2676983

[25] Sam Speight. 2018. Impredicative Encodings of Inductive Types in Homotopy Type Theory. Master's thesis. Retrieved from https://www.cs.ox.ac.uk/people/sam.speight/publications/sams-hott-thesis.pdf

[26] Morten Heine Sørensen and Pawel Urzyczyn. 2006. *Lectures on the Curry-Howard isomorphism*. Elsevier. Retrieved from https://www.sciencedirect.com/bookseries/studies-in-logic-and-the-foundations-of-mathematics/vol/149/suppl/C

[27] The Univalent Foundations Program. 2013. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Retrieved from https://homotopytypetheory.org/book

[28] Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. 2021. Cubical Agda: A dependently typed programming language with univalence and higher inductive types. *Journal of Functional Programming* 31, (2021), e8. https://doi.org/10.1017/S0956796821000034

[29] Niels van der Weide and Herman Geuvers. 2019. The Construction of Set-Truncated Higher Inductive Types. *Electronic Notes in Theoretical Computer Science* 347, (November 2019), 261–280. https://doi.org/10.1016/j.entcs.2019.09.014

[30] Niels van der Weide. 2020. Constructing Higher Inductive Types as Groupoid Quotients. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '20)*, July 2020. ACM, 929–943. https://doi.org/10.1145/3373718.3394803