The Construction of Set-Truncated Higher Inductive Types

Niels van der Weide [1,2]

Institute for Computation and Information Sciences
Radboud Universiteit
Nijmegen, The Netherlands

Herman Geuvers[3]

Institute for Computation and Information Sciences
Radboud Universiteit
Nijmegen, The Netherlands

Abstract

We construct finitary set-truncated higher inductive types (HITs) from quotients and the propositional truncation. For that, we first define signatures as a modification of the schema by Basold *et al.*, and we show they give rise to univalent categories of algebras in both sets and setoids. To interpret HITs, we use the well-known method of initial algebra semantics. The desired algebra is obtained by lifting the quotient adjunction to the level of algebras and adapting Dybjer's and Moeneclaey's interpretation of HITs in setoids. From this construction, we conclude that the equality types of HITs are freely generated and that HITs are unique. The results are formalized in the UniMath library.

Keywords: higher inductive types, homotopy type theory, category theory, setoids, intuitionistic type theory, Coq

1 Introduction

Homotopy type theory (HoTT) is a form of intensional type theory. It has semantics in the simplicial sets model [28], and types represent spaces, terms represent points, and equalities represent paths, and so on. Furthermore, equality is proof relevant and we frequently talk about homotopies: paths between paths.

One of the main features of HoTT is higher inductive types (HITs) [41]. These are types generated by constructors for their points, paths, homotopies, and so on. HITs have been used in numerous applications among which are homotopical patch theory [10], synthetic homotopy theory [31,32,41], defining type theory within type theory [6,7], constructive finiteness [22], and the partiality monad [5]. To get a feeling for what HITs are, let us look at some examples.

`Inductive` $S^1 :=$	`Inductive` $\|A\| :=$	`Inductive` $\mathbb{Z}_2 :=$
$\mid$ **base** : S^1	$\mid \|\cdot\| : A \to \|A\|$	$\mid$ **Z** : $\mathbb{Z}_2$
$\mid$ **loop** : **base** = **base**	$\mid p : \prod(x, y : \|A\|), x = y$	$\mid$ **S** : $\mathbb{Z}_2 \to \mathbb{Z}_2$
		$\mid$ **mod** : $\prod(x : \mathbb{Z}_2), \mathbf{S}(\mathbf{S}\ x) = x$
		$\mid$ **Ztrunc** : $\mathsf{isaset}(\mathbb{Z}_2)$

The first one, S^1, is the *circle*. It has a point **base** and a path **loop** : **base** = **base**. Since neither the point nor path constructors uses arguments from S^1, this HIT is *non-recursive*. The second, $\|A\|$, is the *propositional*

[1] The authors would like to thank Dan Frumin and Andrej Bauer for inspiring discussions. The authors also thank the reviewers for their helpful comments to improve the readability of this paper. Work on this article was supported by a grant from the COST Action EUTypes CA15123.

[2] Email: nweide@cs.ru.nl

[3] Email: herman@cs.ru.nl

truncation of A. This type is A with all its points identified. Note that the path constructor p uses arguments from $\|A\|$, which means this HIT is *recursive*. The last one, $\mathbb{Z}_2$, is the *integers modulo 2* and note that both the point constructor $\mathbf{S}$ and path constructor **mod** are recursive. A HIT is called *finitary* if its point constructor is described by a finitary polynomial and it is called *set-truncated* if it is a set.

In this paper, we show that all finitary set-truncated HITs can be constructed with quotients and propositional truncations. This briefly means that all higher inductive types exist if a small number of simple ones exist. The main idea of the proof is to take advantage that the quotient is left adjoint functor from setoids to sets [38]. To relate this to HITs, we first define schemes, which are an internalized version of the schema by Basold *et al.* [12], and categories of algebras on them. Then we lift the quotient adjunction to an adjunction from algebras in setoids to algebras in sets. Since initiality implies induction, it suffices to construct the initial setoid algebra for which we adapt the construction by Dybjer and Moeneclaey [20,36]. Note that from our construction we can conclude that all finitary set-truncated HITs can be constructed from non-recursive ones. Since quotients can be constructed from coequalizers and set truncations, we only need coequalizers and propositional/set truncations for this construction [37].

1.1 Background and Related Work

The HITs we study in this paper, are set-truncated and recursion is allowed for both the point and path constructors. It is a variation of the scheme by Basold *et al.* [12], which does not allow constructors for higher paths. Other schemata of higher inductive types have already been defined. *W-suspensions*, developed by Sojakova [39], allow defining HITs without recursive constructors, but these are not necessarily truncated. The scheme by Dybjer and Moenclaey is similar to the one by Basold *et al.*, as it allows recursion for both the points and paths, but in addition, it supports constructors for homotopies. Since types are ω-groupoids [42,33], a type-theoretical version of these also provide a semantic specification of HITs [24]. At the moment, the definition which encompasses all these options, are higher inductive-inductive types (HIITs) [26] where induction-induction and paths in arbitrary dimensions are permitted. These are a generalization of quotient inductive-inductive types (QIITs) developed by Altenkirch *et al.* [4]. Furthermore, Cavallo and Harper extend computational cubical type theory with a scheme for indexed cubical inductive types [9,15].

In addition to the schemata, some meta theory also has been developed. Here we internally provide initial algebra semantics for our scheme meaning that we show initial implies induction. Awodey *et al.* showed a stronger result for inductive types in intensional type theory [11,19], namely that initiality is equivalent to induction, and Sojakova proved this for W-suspensions [39]. For QIITs, initial algebra semantics has also been given [4]. Beside initial algebra semantics, Cavallo and Harper shows that computational cubical type theory, with cubical inductive types, satisfies canonicity [9,15].

The main result of this paper is about constructing a class of higher inductive types from simple ones. Both Kraus and Van Doorn show that the propositional truncation can be constructed from non-recursive HITs [29,43] and Rijke shows that every truncation can be constructed via non-recursive HITs [37]. Note that these all are about truncations instead of a more general scheme. However, contrary to our result, they do not rely on the restriction to set-truncated types. Kaposi *et al.* [27] show that can finitary QIITs can be constructed from one specific QIIT, but their construction relies on UIP. From our construction, we deduce that the path space of each HIT is freely generated. Kraus and Von Raumer obtained a similar property for the coequalizer by providing a nicer induction principle for its path types [30].

Reducing the existence of HITs to the existence of a small number of them, gives an approach to defining the semantics of HITs. Other approaches to this problem have also been considered. Coquand *et al.* interpret various higher inductive types, such as spheres, the torus, the truncation, and the pushout, in cubical type theory [13,17,18]. All of these examples are not set-truncated and thus not covered by our scheme. On the other hand, Lumsdaine and Shulman also study the semantics of higher inductive types using a semantic scheme (cell monads with parameters) and they prove existence in sufficiently nice Quillen model categories [34]. Dybjer and Moeneclaey give an interpretation of their scheme in the groupoid model [20,25].

1.2 Overview

We start in Section 2 by recalling some definitions from HoTT and category theory required for the remainder of the paper. In Section 3, we define signatures for higher inductive types and give a couple of examples. Next we define algebras in both the categories of sets and setoids for such signatures in Section 4. We define the induction principle of HITs via displayed algebras in Section 5 and we show that initial objects satisfy this principle. Section 6 contains the main result of this paper and there we construct an adjunction between algebras in sets and setoids, and we use it to construct the initial algebra in sets via the initial algebra in setoids. We study the consequences of this construction in Section 7 and in Section 8, we conclude and discuss further work.

All material in this paper are formalized over the UniMath library [44]. The proof of the main theorem

is 4060 lines of code and the additional examples are 3278 lines of code. The formalization can be found on https://github.com/nmvdw/SetHITs.

2 Preliminaries

We start by recalling some definitions and notations from homotopy type theory and the basics of category theory in HoTT [2,35,41]. The first notion we need, is the *dependent equality type*.

Definition 2.1 Given a type X, inhabitants $x, y : X$, a type family Y on X, and a path $p : x = y$, we define by path induction a function $\mathsf{transport}^Y\ p : Y\ x \to Y\ y$, which is the identity function for the reflexivity path. If we have a path $p : x = y$ and inhabitants $z_1 : Y\ x$ and $z_2 : Y\ y$, we write $z_1 =_p z_2$ for $\mathsf{transport}^Y\ p\ z_1 = z_2$.

One of the core features of HoTT, is that equality is proof relevant. This means that not all inhabitants of $x = y$ are necessarily equal. Some types might actually have proof irrelevant equality, and we call such types *sets*. More precisely, we define

Definition 2.2 A type X is a *(mere) proposition* if for all $x, y : X$ we have $x = y$. A type X is a *set* if for all $x, y : X$ the type $x = y$ is a proposition. We write $\mathsf{isaprop}(X)$ and $\mathsf{isaset}(X)$ to say that X is a proposition and set respectively.

Our goal is to construct all higher inductive types from two specific ones, namely the *quotient type* and the *propositional truncation*. We only give their introduction rules here, and for their elimination and computation rules, we refer the reader to the literature [41].

Definition 2.3 Let X be a type and let R be an equivalence relation on X. The *quotient type* X/R is the higher inductive type generated by

$$\frac{x : X}{\mathbf{class}\ x : X/R} \qquad \frac{x, y : X \qquad r : R\ x\ y}{\mathbf{classeq}\ r : \mathbf{class}\ x = \mathbf{class}\ y} \qquad \mathsf{isaset}(X/R)$$

Definition 2.4 Let X be a type. The *propositional truncation* $\|X\|$ is the higher inductive type generated by

$$\frac{x : X}{|x| : \|X\|} \qquad \mathsf{isaprop}(\|X\|)$$

Next we briefly discuss category theory in univalent foundations [2,41]. We start by recalling the definition of categories. This is almost the same as the usual definition in mathematics, but there is a slight discrepancy between the objects and arrows. While the objects can be any type, the arrows have to form a set. Proof relevant equality on arrows would induce the structure of a higher category instead of an ordinary one.

Definition 2.5 A *category* $\mathcal{C}$ consists of a type $\mathcal{C}_0$ of objects and a set $X \to Y$ of morphisms for each $X, Y : \mathcal{C}_0$ together with identity morphisms $\mathsf{id}_X : X \to X$ for each $X : \mathcal{C}_0$ and a compositions $g \circ f : X \to Z$ for all mophisms $f : X \to Y$ and $g : Y \to Z$ such that the usual associativity and identity laws holds.

We define isomorphisms in categories the usual way and we denote the type of isomorphisms from X to Y by $X \cong Y$. In the remainder, we also make use of *univalent categories*. These are categories in which equality on objects is equivalent to isomorphisms between them. For the definition of equivalence of equivalences, we refer the reader to the literature [41]. More precisely, we define

Definition 2.6 Let $\mathcal{C}$ be a category. Note that for all objects X and Y we have a map $\mathsf{idtoiso}_{X,Y} : X = Y \to X \cong Y$ sending the reflexivity path to the identity isomorphism. Then we say $\mathcal{C}$ is *univalent* if $\mathsf{idtoiso}_{X,Y}$ is an equivalence for each X and Y.

The primary example of a univalent category is hSet, whose objects are sets and morphisms are functions. Another example of a univalent category which we use frequently, is the category of *setoids*.

Definition 2.7 A *setoid* is a set X together with an equivalence relation on X. A *setoid morphism* between two setoids is a map between the underlying sets which preserves the equivalence relation. The *category* Setoid *of setoids* is the category with setoids and setoid morphisms as objects and morphisms.

Recall that, If R is an equivalence relation on X, each $R\,x\,y$ has to be a proposition. Note that in the usual category of setoids, one would take a quotient of the setoid morphisms. However, we refrain to do so, because in our construction, we do not need any additional equality. If X is a set and R is an equivalence relation on X, then we write (X, R) for the setoid with this data. If no confusion arises, we write $x \equiv y$ for $R\,x\,y$.

We finish this section by giving some operations of functors and natural transformations, which we need in Definition 4.2. Functors and natural transformations are defined the usual way [35], and we write $\mathcal{C} \longrightarrow \mathcal{D}$ and $F \Longrightarrow G$ for the type of functors from $\mathcal{C}$ to $\mathcal{D}$ and transformations from F to G respectively.

Notation 2.8 *We have the following functors.*

- *For each category $\mathcal{C}$, we have the identity $\mathsf{id}_{\mathcal{C}} : \mathcal{C} \longrightarrow \mathcal{C}$.*
- *For $F : \mathcal{C}_1 \longrightarrow \mathcal{C}_2$ and $G : \mathcal{C}_2 \longrightarrow \mathcal{C}_3$, we have a composition $G \circ F : \mathcal{C}_1 \longrightarrow \mathcal{C}_3$;*
- *If $\mathcal{D}$ has binary products, then for $F, G : \mathcal{C} \longrightarrow \mathcal{D}$, we have a product $F \times G : \mathcal{C} \longrightarrow \mathcal{D}$;*
- *If $\mathcal{D}$ has binary sums, then for $F, G : \mathcal{C} \longrightarrow \mathcal{D}$, we have a sum $F + G : \mathcal{C} \longrightarrow \mathcal{D}$;*
- *For each object $X : \mathcal{D}$, we have a constant functor $\mathsf{C}_X : \mathcal{C} \longrightarrow \mathcal{D}$.*

Notation 2.9 *We have the following natural transformations.*

- *Given $F, G : \mathcal{C} \longrightarrow \mathcal{D}$, we have*

$$\mathsf{inl} : F \Longrightarrow F + G \qquad \mathsf{inr} : G \Longrightarrow F + G \qquad \mathsf{pr}_1 : F \times G \Longrightarrow F \qquad \mathsf{pr}_2 : F \times G \Longrightarrow G$$

- *Given two transformations $\eta_1 : G_1 \circ F \Longrightarrow G_2 \circ F$ and $\eta_2 : G_1 \circ F \Longrightarrow G_3 \circ F$, we have a pairing*

$$(\eta_1, \eta_2) : G_1 \circ F \Longrightarrow G_2 \times G_3 \circ F.$$

- *Given functors $F : \mathcal{C}_1 \longrightarrow \mathcal{C}_2$ and $G_1.G_2 : \mathcal{C}_2 \longrightarrow \mathcal{C}_3$, and a transformation $\eta : G_1 \Longrightarrow G_2$, we have*

$$\eta \rhd F : G_1 \circ F \Longrightarrow G_2 \circ F$$

- *Given a functor $F : \mathcal{C} \longrightarrow \mathrm{hSet}$, a set $X : \mathrm{hSet}$, and $x : X$, we have a transformation $\mathsf{c}_x : F \Longrightarrow \mathsf{C}_X$*

The type of (η_1, η_2) might seem unnatural. However, it is precisely this type we need for the constructions in Definition 4.2..

3 Signature of HITs

Before we study the construction of set truncated HITs, we first give a precise definition of those. We describe HITs by saying how to construct points and paths. For example, the integers modulo 2, which we considered in the introduction, has two constructors for points, namely **Z** ad **S**, and one for the paths, namely **mod**. The signature of this type would thus indicate that we have one nullary and one unary operation, and a path of the required type. In this section, we define a type of signatures as internalized variation of the one by Basold *et al.* [12], and then in the upcoming sections, we define HITs for a signature.

The first ingredient of the signature, is the arity of the point constructor. This is described by a *finitary polynomial functor*, and we define those as the following type.

Definition 3.1 We define the type $\mathcal{P}$ of *codes of finitary polynomials* as the inductive type generated by the following constructors.

$$\frac{X : \mathrm{hSet}}{\mathbf{C}\ X : \mathcal{P}} \qquad \mathbf{I} : \mathcal{P} \qquad \frac{P : \mathcal{P} \qquad Q : \mathcal{P}}{P + Q : \mathcal{P}} \qquad \frac{P : \mathcal{P} \qquad Q : \mathcal{P}}{P \times Q : \mathcal{P}}$$

In the next section, we show that each code $P : \mathcal{P}$ gives rise to a functor $[\![P]\!] : \mathrm{hSet} \longrightarrow \mathrm{hSet}$. Each higher inductive type H has a point constructor $c : [\![P]\!]\ H \to H$, which represents the introduction rule for points.

HITs also have an introduction rule for paths, and for those, we need to give the possible endpoints of paths. Each path is given by a universally quantified equation, which can possibly make use of the point constructor. For this reason, the type of endpoint must depend on the data of the point constructor. We also indicate the source and target of the equation, and both of them depend polynomially on the HIT being defined. The type of endpoints is defined inductively.

Definition 3.2 Given polynomials, $A, P, Q : \mathcal{P}$, the type $\mathcal{E}_A(P, Q)$ of *endpoints* is inductively generated by the following constructors.

$$\frac{P : \mathcal{P}}{\mathbf{id}_A : \mathcal{E}_A(P, P)} \qquad \frac{P, Q, R : \mathcal{P} \qquad e_1 : \mathcal{E}_A(P, Q) \qquad e_2 : \mathcal{E}_A(Q, R)}{e_1 \cdot e_2 : \mathcal{E}_A(P, R)}$$

$$\frac{P, Q : \mathcal{P}}{\mathbf{inl} : \mathcal{E}_A(P, P + Q)} \qquad \frac{P, Q : \mathcal{P}}{\mathbf{inr} : \mathcal{E}_A(Q, P + Q)} \qquad \frac{P, Q : \mathcal{P}}{\mathbf{pr}_1 : \mathcal{E}_A(P \times Q, P)} \qquad \frac{P, Q : \mathcal{P}}{\mathbf{pr}_2 : \mathcal{E}_A(P \times Q, Q)}$$

$$\mathbf{constr} : \mathcal{E}_A(A, \mathbf{I}) \qquad \frac{P : \mathcal{P} \qquad X : \mathrm{hSet} \qquad x : X}{\mathbf{c}\, x : \mathcal{E}_A(P, \mathbf{C}\, X)} \qquad \frac{P, Q, R : \mathcal{P} \qquad e_1 : \mathcal{E}_A(P, Q) \qquad e_2 : \mathcal{E}_A(P, R)}{(e_1, e_2) : \mathcal{E}_A(P, Q \times R)}$$

In the formalization, we also include for each $f : X \to Y$ an endpoint $\mathbf{fmap}\, f : \mathcal{E}_A(\mathbf{C}\, X, \mathbf{C}\, Y)$, which we do not discuss here. The polynomial A represents the point constructor and that explains the endpoint **constr**. The polynomials P and Q represent the source and target of the equation respectively. If we have a set X with a map $c : [\![A]\!]\, X \to X$, each endpoint $e : \mathcal{E}_A(P, Q)$ gives rise to a natural morphism from the functor $[\![P]\!]$ to $[\![Q]\!]$ where **constr** is interpreted using c. These maps are the left- and right-hand side of the equations.

Now we put it all together to define signatures of HITs. Note that we index the path constructors by a type meaning that we could possibly have infinitely many path constructors.

Definition 3.3 A *HIT signature* $\mathcal{S}$ consists of

- A polynomial $\mathcal{S}_{\mathsf{pt}} : \mathcal{P}$ representing the point constructor;
- A type $\mathcal{S}_{\mathsf{pth}}$ representing the names of path constructors;
- A family $\mathcal{S}_{\mathsf{arg}} : \mathcal{S}_{\mathsf{pth}} \to \mathcal{P}$ representing the arguments of the paths;
- Maps $\mathcal{S}_{\mathsf{lhs}}, \mathcal{S}_{\mathsf{rhs}} : \prod(j : \mathcal{S}_{\mathsf{pth}}), \mathcal{E}_{\mathcal{S}_{\mathsf{pt}}}(\mathcal{S}_{\mathsf{arg}}\, j, \mathbf{I})$ representing the left- and right-hand side of the equations.

Briefly, the signature $\mathcal{S}$ represents the following HIT

```
Inductive H :=
| c : [[S_pt]] H → H
| p : ∏(j : S_pth), ∏(x : [[S_arg j]] H), [[S_lhs j]] H x = [[S_rhs j]] H x
| s : isaset(H)
```

Since our goal is to interpret set-truncated HITs, we require the constructor s. To illustrate the possibilities of this definition, we define two examples from the introduction as HIT signatures. The first one is the integers modulo 2 and the second one is the propositional truncation of a set.

Example 3.4 Recall $\mathbb{Z}_2$ from the introduction. We represent it by the signature mod defined as follows

$$\mathsf{mod}_{\mathsf{pt}} := \mathbf{I} + (\mathbf{C}\, \mathbf{1}) \qquad \mathsf{mod}_{\mathsf{pth}} := \mathbf{1} \qquad \mathsf{mod}_{\mathsf{arg}} := \mathbf{I}$$

$$\mathsf{mod}_{\mathsf{lhs}}\, j := \mathbf{inl} \cdot \mathbf{constr} \cdot \mathbf{inl} \cdot \mathbf{constr} \qquad \mathsf{mod}_{\mathsf{rhs}}\, j := \mathbf{id}_{\mathbf{I}}$$

Intuitively, this signature represents a HIT H with operation $H + \mathbf{1} \to H$ representing the successor and zero. Note that the endpoint $\mathbf{inl} \cdot \mathbf{constr}$ takes the successor, so, the equation says that $S(S\, x) = x$ for all $x : X$. Since signatures can depend on types, we can also define the propositional truncation.

Example 3.5 Let A be a set. We represent its truncation by the following signature.

$$\mathsf{trunc}_{\mathsf{pt}} := \mathbf{C}\, A \qquad \mathsf{trunc}_{\mathsf{pth}} := \mathbf{1} \qquad \mathsf{trunc}_{\mathsf{arg}} := \mathbf{I} \times \mathbf{I}$$

$$\mathsf{trunc}_{\mathsf{lhs}}\, j := \mathbf{pr}_1 \qquad \mathsf{trunc}_{\mathsf{rhs}}\, j := \mathbf{pr}_2$$

Other examples, such as the integers, free algebras, and the ring of polynomials, can be found in the formalization. This signature represents a HIT H with a map $A \to H$ and for which all inhabitants are equal, which is precisely the propositional truncation from Definition 2.4 for sets A.

4 Algebras

4.1 Algebras in Sets

To define HITs on a signature, we need to give the introduction, elimination, and compuation rules. Let us start with the introduction rule, which we define via algebras. Note that this rule comes in two flavors: one for the points and one forthe paths. For this reason, we define algebras in two steps.

We start by saying how to interpret the point constructors.

Definition 4.1 For each $P : \mathcal{P}$, we define a functor $[\![P]\!] : \mathrm{hSet} \longrightarrow \mathrm{hSet}$ as follows

$$[\![\mathbf{C}\, X]\!] := \mathsf{C}_X \qquad [\![\mathbf{I}]\!] := \mathsf{id}_{\mathrm{hSET}} \qquad [\![P + Q]\!] := [\![P]\!] + [\![Q]\!] \qquad [\![P \times Q]\!] := [\![P]\!] \times [\![Q]\!]$$

We write $\mathrm{FALG}(F)$ for the category of algebras on the functor $F : \mathcal{C} \longrightarrow \mathcal{C}$. Its objects are pairs $X : \mathcal{C}$ together with an arrow $\mathcal{C}_1(F\ X, X)$. The morphisms from (X, f) to (Y, g) consist of maps $h : \mathcal{C}_1(X, Y)$ such that $h \circ f = g \circ F(h)$. Note that we always have a forgetful functor $\mathsf{U}_F : \mathrm{FALG}(F) \longrightarrow \mathcal{C}$.

Now we define the category $\mathrm{PREALG}_{\mathrm{HSET}}(\mathcal{S})$ to be $\mathrm{FALG}(\llbracket \mathcal{S}_{\mathsf{pt}} \rrbracket)$. Since the objects do not satisfy the equations in $\mathcal{S}$, we call this the category of *prealgebras*. To obtain actual algebras of $\mathcal{S}$, we need to interpret the equations. For that, we first give a semantics of endpoints as natural transformations, which are defined using the transformations from Lemma 2.9. Besides, we use for each polynomial P the transformation $\mathsf{constr}_P : \llbracket P \rrbracket \circ \mathsf{U}_P \Longrightarrow \llbracket \mathbf{I} \rrbracket \circ \mathsf{U}_P$ whose components are given by the prealgebra map.

Definition 4.2 For each endpoint $e : \mathcal{E}_A(P, Q)$, we define a natural transformation $\llbracket e \rrbracket : \llbracket P \rrbracket \circ \mathsf{U}_A \Longrightarrow \llbracket Q \rrbracket \circ \mathsf{U}_A$

$$\llbracket \mathbf{id}_P \rrbracket := \mathsf{id} \qquad \llbracket e_1 \cdot e_2 \rrbracket := \llbracket e_2 \rrbracket \circ \llbracket e_1 \rrbracket \qquad \llbracket \mathbf{inl} \rrbracket := \mathsf{inl} \rhd \mathsf{U}_A \qquad \llbracket \mathbf{inr} \rrbracket := \mathsf{inr} \rhd \mathsf{U}_A \qquad \llbracket \mathbf{c}\ t \rrbracket := \mathsf{c}_t$$

$$\llbracket \mathbf{pr}_1 \rrbracket := \mathsf{pr}_1 \rhd \mathsf{U}_A \qquad \llbracket \mathbf{pr}_2 \rrbracket := \mathsf{pr}_2 \rhd \mathsf{U}_A \qquad \llbracket (e_1, e_2) \rrbracket := (\llbracket e_1 \rrbracket, \llbracket e_2 \rrbracket) \qquad \llbracket \mathbf{constr} \rrbracket := \mathsf{constr}_A$$

Note that $\llbracket \mathbf{constr} \rrbracket$ is required to have type $\llbracket P \rrbracket \circ \mathsf{U}_P \Longrightarrow \llbracket \mathbf{I} \rrbracket \circ \mathsf{U}_P$. Since the functors $\llbracket \mathbf{I} \rrbracket \circ \mathsf{U}_P$ and U_P are only equal up to propositional equality, we defined constr_A to be of that type instead of the expected $\llbracket P \rrbracket \circ \mathsf{U}_P \Longrightarrow \mathsf{U}_P$, Now we have everything in place to define algebras on $\mathcal{S}$. An algebra on $\mathcal{S}$ consists of a prealgebra together with proofs that the equations in $\mathcal{S}$ are satisfied. Since the carrier of each prealgebra is a set, the equations form a proposition. Hence, we define the category of *$\mathcal{S}$-algebras* as a full subcategory of $\mathrm{PREALG}_{\mathrm{HSET}}(\mathcal{S}_{\mathsf{pt}})$.

Definition 4.3 Let $\mathcal{S}$ be a HIT signature. Then we define the category $\mathrm{ALG}_{\mathrm{HSET}}(\mathcal{S})$ of *$\mathcal{S}$-algebras* as the full subcategory of $\mathrm{PREALG}_{\mathrm{HSET}}(\mathcal{S}_{\mathsf{pt}})$ such that each object X satisfies

$$\prod (j : \mathcal{S}_{\mathsf{pth}}), \prod (x : \llbracket \mathcal{S}_{\mathsf{arg}}\ j \rrbracket\ X), \llbracket \mathcal{S}_{\mathsf{lhs}}\ j \rrbracket\ X\ x = \llbracket \mathcal{S}_{\mathsf{rhs}}\ j \rrbracket\ X\ x.$$

For an algebra X, we denote its operation by $\mathsf{point}_X : \llbracket \mathcal{S}_{\mathsf{pt}} \rrbracket\ X \to X$. The path witnessing the equalities of the algebra is denoted by $\mathsf{path}_X : \prod (j : \mathcal{S}_{\mathsf{pth}}), \prod (x : \llbracket \mathcal{S}_{\mathsf{arg}}\ j \rrbracket\ X), \llbracket \mathcal{S}_{\mathsf{lhs}}\ j \rrbracket\ x = \llbracket \mathcal{S}_{\mathsf{rhs}}\ j \rrbracket\ x$. Note that the category of $\mathcal{S}$-algebras is univalent. This follows from the fact that the category of algebras on a functor is univalent and that univalence is preserved under taking full subcategories.

Proposition 4.4 *The category of $\mathcal{S}$-algebras in sets is univalent.*

Before we look at algebras in setoids in more detail, we recall the examples in the previous section and look what algebras on those signatures are.

Example 4.5 Recall the signature mod from Example 3.4. A prealgebra of mod consists of a set X and a map $f : X + \mathbf{1} \to X$. We define $S_X\ x := f(\mathsf{inl}\ x)$ and $Z_X = f(\mathsf{inr}\ \mathbf{tt})$. An algebra of mod consists of a prealgebra X such that for all $x : X$, we have $S_X(S_X\ x) = x$.

Example 4.6 Recall trunc from Example 3.5. For a set A. an algebra of $\mathsf{trunc}\ A$ consists of a set X and a map $f : A \to X$ such that for all $(x, y) : X \times X$, we have $x = y$. In particular, this means X is a proposition.

4.2 Algebras in Setoids

Our goal is to construct HITs as a quotient of a certain setoids. To guarantee that the resulting quotient has the right introduction rule, we require extra structure from the setoid. This is given by an algebra structure on the setoid.

To interpret the action of P on setoids, we first define its action on equivalence relations, and we need some preliminary operations for that. Note that for each set T, we have an equivalence relation $\mathsf{Path}\ T$ on T such that $\mathsf{Path}\ T\ x\ y := x = y$. Furthermore, given types X and Y with equivalence relations R_X and R_Y on them, we can define equivalence relations $R_X + R_Y$ and $R_X \times R_Y$ on $X + Y$ and $X \times Y$ respectively. These are defined as follows

$$R_X \times R_Y\ (x_1, y_1)\ (x_2, y_2) := (R_X\ x_1\ x_2) \times (R_Y\ y_1\ y_2)$$

$$R_X + R_Y\ (\mathsf{inl}\ x_1)\ (\mathsf{inl}\ x_2) := R_X\ x_1\ x_2 \qquad R_X + R_Y\ (\mathsf{inr}\ y_1)\ (\mathsf{inr}\ y_2) := R_Y\ y_1\ y_2$$

$$R_X + R_Y\ (\mathsf{inl}\ x)\ (\mathsf{inr}\ y) := \mathbf{0} \qquad R_X + R_Y\ (\mathsf{inr}\ y)\ (\mathsf{inl}\ x) := \mathbf{0}$$

Definition 4.7 Let R be an equivalence relation on a set X and let $P : \mathcal{P}$ be a polynomial. By induction, we define an equivalence relation $\widehat{P}\ R$ on $\llbracket P \rrbracket\ X$.

$$\widehat{(\mathbf{C}\ T)}\ R := \mathsf{Path}\ T \qquad \widehat{\mathbf{I}}\ R := R \qquad \widehat{(P + Q)}\ R := (\widehat{P}\ R) + (\widehat{Q}\ R) \qquad \widehat{(P \times Q)}\ R := (\widehat{P}\ R) \times (\widehat{Q}\ R)$$

Now we define the functor $\langle P \rangle : \textsc{Setoid} \longrightarrow \textsc{Setoid}$ by $\langle P \rangle (X, R) = (\llbracket P \rrbracket\, X, \widehat{P}\, R)$.

The functor $\langle P \rangle$ could also be defined by using that the category of setoids is Cartesian closed. However, we chose this definition, because it was more convenient in the formalization. This is because the underlying set of $\langle P \rangle (X)$ can be computed definitionally.

With these definitions in place, we obtain a category $\textsc{PreAlg}_{\textsc{Setoid}}(\mathcal{S})$ of *setoid prealgebras on* $\mathcal{S}$. To define algebras on $\mathcal{S}$, we also need to interpret endpoints for which we use setoid morphisms instead of natural transformations.

Definition 4.8 Let $e : \mathcal{E}_A(P, Q)$ be an endpoint and let X be a setoid prealgebra on A. Then we define $\langle e \rangle$ to be the setoid morphism from $\langle P \rangle (X)$ to $\langle Q \rangle (X)$ whose carrier is $\llbracket e \rrbracket$.

The requirement for $\langle e \rangle$ to be a setoid morphism, is that for $x, y : \langle P \rangle X$ with $r : x \equiv y$, we have $\llbracket e \rrbracket x \equiv \llbracket e \rrbracket y$. The category of algebras on $\mathcal{S}$ is defined differently for setoids than for sets. While for sets, the equations of algebras are witnessed by actual equalities, the equations for setoids are witnessed by the equivalence relation. Note that such relations are families of propositions meaning again this gives rise to a proposition and thus we define it as a full subcategory.

Definition 4.9 Let $\mathcal{S}$ be a HIT signature. Then we define the category of $\mathcal{S}$*-setoid-algebras* as the full subcategory of $\textsc{PreAlg}_{\textsc{Setoid}}(\mathcal{S}_{\mathsf{pt}})$ such that each object satisfies

$$\prod (j : \mathcal{S}_{\mathsf{pth}}), \prod (x : \mathcal{S}_{\mathsf{arg}}\, j), \llbracket \mathcal{S}_{\mathsf{lhs}}\, j \rrbracket\, x \equiv \llbracket \mathcal{S}_{\mathsf{rhs}}\, j \rrbracket\, x.$$

Since the category of setoids is univalent, the category of $\mathcal{S}$-setoid-algebras is univalent as well.

Proposition 4.10 *The category of* $\mathcal{S}$*-setoid-algebras is univalent.*

5 The Induction Principle

With the introduction rules covered by the algebra structure, we now take a look a the elimination and computation rules. For this, we use *displayed algebras* [26,39]. These represent the input of the elimination rule. For the output, we define *displayed algebra maps*, also known as sections. The elimination rule says that we have a displayed algebra map to every displayed algebra, while the computation rule says that the algebra in place. Once we have this machinery in place, we define *higher inductive type* on a signature.

Since our goal is to construct HITs, we need to find an algebra for which the elimination rule holds, and for that, we use initial algebra semantics [4,11,39]. More specifically, we show that the initial algebra satisfies the induction rule. Hence, to obtain a HIT, it suffices to construct the initial algebra, which is more convenient in the language of category theory.

5.1 Displayed Algebras

A displayed algebra is the input of the elimination rule. This means that we have a dependent family and a dependent map over the point constructor. Furthermore, dependent versions of the equations in the signature need to hold. Displayed algebras are similar to displayed categories [3].

To formulate these requirements precisely, we need two preliminary definitions. The first one is the action of polynomials on families of sets while the second one interprets endpoints as dependent maps.

Definition 5.1 Given are $P : \mathcal{P}$ and $Y : X \to \textsc{hSet}$. We define $\overline{P}\, Y : \llbracket P \rrbracket\, X \to \textsc{hSet}$ by induction

$$\overline{\mathbf{C}\, X}\, Y\, x := X \qquad \overline{\mathbf{I}}\, Y\, x := Y\, x \qquad \overline{P \times Q}\, Y\, x := \overline{P}\, Y\, (\mathsf{pr}_1\, x) \times \overline{Q}\, Y\, (\mathsf{pr}_2\, x)$$

$$\overline{P + Q}\, Y\, (\mathsf{inl}\, x) := \overline{P}\, Y\, x \qquad \overline{P + Q}\, Y\, (\mathsf{inr}\, x) := \overline{Q}\, Y\, x$$

Definition 5.2 Let A be a polynomial, X be a prealgebra on A, and let $e : \mathcal{E}_A(P, Q)$. Suppose, we have a family Y on X and a map $c : \prod (z : \llbracket A \rrbracket X), \overline{P} Y z \to Y\, (\mathsf{point}_X\, z)$. We define a map $\overline{e} c : \prod (z : \llbracket P \rrbracket X), \overline{P} Y z \to \overline{Q} Y (\llbracket e \rrbracket z)$ by induction on e

$$\overline{\mathbf{id}_P}\, c\, z\, y := y \qquad \overline{e_1 \cdot e_2}\, c\, z\, y := \overline{e_2}\, c\, (\llbracket e_1 \rrbracket\, X\, z)\, (\overline{e_1}\, c\, z\, y) \qquad \overline{\mathbf{inl}}\, c\, z\, y := y \qquad \overline{\mathbf{inr}}\, c\, z\, y := y \qquad \overline{\mathbf{c}\, t}\, c\, z\, y := t$$

$$\overline{\mathbf{pr}_1}\, c\, z\, y := \mathsf{pr}_1\, y \qquad \overline{\mathbf{pr}_2}\, c\, z\, y := \mathsf{pr}_2\, y \qquad \overline{(e_1, e_2)}\, c\, z\, y := (\overline{e_1}\, c\, z\, y, \overline{e_2}\, c\, z\, y) \qquad \overline{\mathbf{constr}}\, c := c$$

With this in place, we define displayed algebras. Note that since we are working in a family of sets, we need to use the dependent equality type from Definition 2.1 instead of the ordinary one.

Definition 5.3 Give are a signature $\mathcal{S}$ and an algebra X on $\mathcal{S}$. Then a *displayed algebra* over X consists of

- A type family $Y : X \to \text{hSet}$;
- An operation $c_Y : \prod(z : [\![\mathcal{S}_{\mathsf{pt}}]\!] X), \overline{\mathcal{S}_{\mathsf{pt}}}\, Y\, z \to Y\, (\mathsf{point}_X\, z)$;
- For each $j : \mathcal{S}_{\mathsf{pth}}$, $x : [\![\mathcal{S}_{\mathsf{arg}}]\!]\, X$, and $y : \overline{\mathcal{S}_{\mathsf{arg}}}\, Y\, x$, a path $p_Y : \overline{\mathcal{S}_{\mathsf{lhs}}}\, c_Y\, x\, y =_{\mathsf{path}_X\, j\, x} \overline{\mathcal{S}_{\mathsf{rhs}}}\, c_Y\, x\, y$.

Now we got the input for the elimination rule in place and the next step is to look at the output. This is a dependent map which preserves the algebra structure. To state this preservation property, we need yet another operation on polynomials.

Definition 5.4 Let P be a polynomial, let X be a set, and let Y be a family of sets on X. Given a map $f : \prod(x : X), Y\, x$, we define a map $\overline{P}\, f : \prod(x : [\![P]\!]\, X), \overline{P}\, Y\, x$ by induction

$$\overline{\mathbf{C}\, X}\, f\, x := x \qquad \overline{\mathbf{I}}\, f := f \qquad \overline{P \times Q}\, f\, x := (\overline{P}\, f\, (\mathsf{pr}_1\, x), \overline{Q}\, f\, (\mathsf{pr}_2\, x))$$

$$\overline{P + Q}\, f\, (\mathsf{inl}\, x) := \overline{P}\, f\, x \qquad \overline{P + Q}\, f\, (\mathsf{inr}\, x) := \overline{Q}\, f\, x$$

Definition 5.5 Let Y be a displayed algebra over X. Then a *displayed algebra map* to Y consists of a map $f : \prod(x : X), Y\, x$ such that for each $x : [\![\mathcal{S}_{\mathsf{pt}}]\!] X$, we have $f(\mathsf{point}_X\, x) = c_Y\, x\, (\overline{\mathcal{S}_{\mathsf{pt}}}\, f\, x)$.

With all this in place, we can define the notion of a *higher inductive type* on a signature. Note that a HIT needs to be an algebra so that we have the correct introduction rules and note that we use displayed algebras to formulate the elimination principle.

Definition 5.6 Let $\mathcal{S}$ be a HIT signature. A *higher inductive type* on $\mathcal{S}$ consists of an algebra H such that for each displayed algebra Y on H, we have a displayed algebra map to Y.

Note that HITs of $\mathcal{S}$ satisfy analogous rules to those defined by Basold *et al.* [12]. As usual, the induction rule only guarantees the existence of a dependent map. This obtained map is unique, which is again proven by induction. Furthermore, HITs on the signatures we discussed before, the integers modulo 2 in Example 3.4 and the propositional truncation in Example 3.5, satisfy the induction rule given in Basold *et al.* [12].

There are only two differences: the computation rule only holds propositionally rather than definitionally. In addition, we can only map HITs into sets. This is because it also has constructors, which guarantee it is a set. The algebra structure gives the right introduction rules and the existence of the displayed algebra map gives the required elimination and computation rules.

5.2 Obtaining Induction from Initiality

Next we show how to obtain the induction principle from initiality. This way it suffices construct an initial algebra of the signature to obtain a HIT, which is more convenient in category theory. To this end, we first define the total algebra $\int Y$ of a displayed Y on X together a projection π_1^Y to X.

We define the carrier of $\int Y$ by the dependent sum $\sum(x : X), Y\, x$ and for brevity, we denote this by $\int Y$. To show this is an algebra, we first need to define a map $c_{\int Y} : [\![\mathcal{S}_{\mathsf{pt}}]\!]\, (\int Y) \to \int Y$. The main idea is that we use the algebra map of X for the first component and the displayed algebra map of Y for the second one. We introduce an intermediate definition, which allows us to access the right data.

Definition 5.7 Let P be a polynomial and let Y be a family of sets on X. We define a map

$$\pi_1^P : [\![P]\!]\, (\textstyle\sum(x : X), Y\, x) \to [\![P]\!]\, X \qquad \pi_1^P\, x := [\![P]\!]\, \mathsf{pr}_1\, x$$

We also define a map $\pi_2^P : \prod(x : [\![P]\!]\, (\sum(x : X), Y\, x)), \overline{P}\, Y\, (\pi_1^P\, x)$ by induction on P.

$$\pi_2^{\mathbf{C}\, T}\, x := x \qquad \pi_2^{\mathbf{I}}\, x := \mathsf{pr}_2\, x \qquad \pi_2^{P \times Q}\, x := (\pi_2^P\, (\mathsf{pr}_1\, x), \pi_2^Q\, (\mathsf{pr}_2\, x))$$

$$\pi_2^{P+Q}\, (\mathsf{inl}\, x) := \pi_2^P\, x \qquad \pi_2^{P+Q}\, (\mathsf{inr}\, x) := \pi_2^Q\, x$$

Now let $z : [\![\mathcal{S}_{\mathsf{pt}}]\!]\,(\int Y)$. The first coordinate of $\mathsf{point}_{\int Y}\, z$ is defined to be $\mathsf{point}_X(\pi_1^{\mathcal{S}_{\mathsf{pt}}}\, x)$. The second coordinate of $\mathsf{point}_{\int Y}\, z$ is defined by $c_Y\,(\pi_1^{\mathcal{S}_{\mathsf{pt}}}\, x)\,(\pi_2^{\mathcal{S}_{\mathsf{pt}}}\, x)$. The main challenge lies within proving the equations. To do so, we compute the valuation of endpoints in the total algebra via those in X and Y.

Lemma 5.8 *For every endpoint $e : \mathcal{E}_{\mathcal{S}_{pt}}(P, Q)$, we have*

$$\mathsf{epr}_1 : [\![e]\!]\; X\; (\pi_1^P\; x) = \pi_1^Q([\![e]\!]\;(\int Y)\; x) \qquad \mathsf{epr}_2 : \overline{e}\; c_Y\; (\pi_2^P\; x) =_{\mathsf{epr}_1} \pi_2^Q\; ([\![e]\!]\;(\int Y)\; x)$$

All in all, given a displayed algebra Y on some algebra X, we get the *total algebra* $\int Y$. We also define an algebra homomorphism $\pi_1^Y : \int Y \to X$, which sends z to its first projection $\mathsf{pr}_1\; z$. The reason why we care about this construction, is the following proposition.

Proposition 5.9 *From an algebra homomorphism $f : X \to \int Y$ such that $\pi_1^Y \circ f$ is the identity, we obtain a displayed algebra map from X to Y.*

Now suppose X is an initial object in the category of algebras. Then, due to initiality, we always have a map $f : X \to \int Y$, and the composition $\pi_1^Y \circ f : X \to X$ must be the identity because initiality ensures uniqueness of homomorphisms. Hence, we conclude

Corollary 5.10 *If X is an initial object in the category of algebras on $\mathcal{S}$, then X is a HIT for $\mathcal{S}$.*

6 Constructing the Initial Algebra

To construct the initial algebra, we first lift the quotient to a left adjoint functor from algebras in setoids to algebras in sets. Such functors preserve initial objects, and thus it suffices to construct the initial setoid algebra. For that, we use an adaption of Dybjer's and Moeneclaey's interpretation of HITs in the setoid model [20,36]. In the remainder of this section, we work with a fixed signature $\mathcal{S}$.

6.1 Quotient Adjunction

We start by defining a left adjoint functor $\mathcal{Q} : \textsc{Setoid} \longrightarrow \textsc{hSet}$. On objects, we define $\mathcal{Q}\,(X, R) = X/R$ and its action on morphisms is defined by recursion on the quotient type. The laws are proven by quotient induction. The right adjoint is the path setoid functor $\pi_0 : \textsc{hSet} \longrightarrow \textsc{Setoid}$. For sets X, we define the functor $\pi_0\; X$ to be $(X, \mathsf{Path}\; X)$. Note that this action is functorial and that this gives rise to an adjunction.

Lemma 6.1 (Theorem 2.20 from [38]) *We have an adjunction $\mathcal{Q} \dashv \pi_0$.*

6.2 The Adjunction on Prealgebras

With this adjunction in place, our first step is to lift it to the level of prealgebras. For this, we use a result from Hermida and Jacobs [23]. Note that this is also related to the fact that the 2-functor from the bicategory of endofunctors to the bicategory of categories preserves adjunctions [40].

The first step, is to lift the functors for which we use the following lemma.

Lemma 6.2 *Given are categories $\mathcal{C}$ and $\mathcal{D}$, functors $A_1 : \mathcal{C} \longrightarrow \mathcal{C}$, $A_2 : \mathcal{D} \longrightarrow \mathcal{D}$, and $F : \mathcal{C} \longrightarrow \mathcal{D}$, and a natural transformation $n : A_2 \circ F \Longrightarrow F \circ A_1$. Then we get a functor $F^{\textsc{PreAlg}} : \textsc{FAlg}(A_1) \longrightarrow \textsc{FAlg}(A_2)$.*

For algebras $(X, f) : \textsc{FAlg}(A_1)$, we define $F^{\textsc{PreAlg}}(X, f) = (F\; X, F\; f \circ n\; X)$. To lift the quotient and the path setoid, we need a lemma.

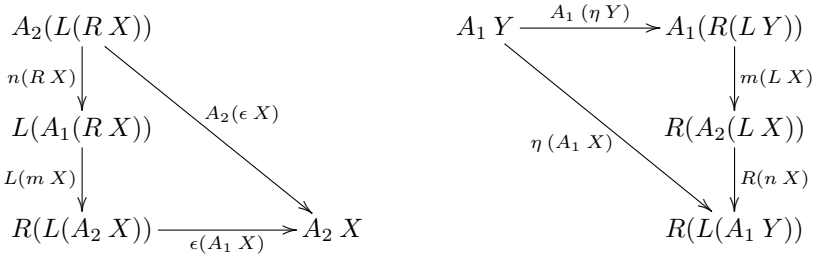
Lemma 6.3 *The functors π_0 and $\mathcal{Q}$ commute with sums and products. This gives rise to two natural isomorphisms $n_1^P : \langle P \rangle \circ \pi_0 \Longrightarrow \pi_0 \circ [\![P]\!]$ and $n_2^P : \langle P \rangle \circ \mathcal{Q} \Longrightarrow \mathcal{Q} \circ [\![P]\!]$.*

Proving that $\mathcal{Q}$ commutes with products makes essential use of the double recursion principle of the quotient type, which allows defining functions $\mathcal{Q}\; X \times \mathcal{Q}\; Y \to Z$. Using Lemmata 6.2 and 6.3, we lift both the quotient and the path setoid functors to obtain.

$$\mathcal{Q}^{\textsc{PreAlg}} : \textsc{PreAlg}_{\textsc{Setoid}}(\mathcal{S}) \longrightarrow \textsc{PreAlg}_{\textsc{hSet}}(\mathcal{S}) \qquad \pi_0^{\textsc{PreAlg}} : \textsc{PreAlg}_{\textsc{hSet}}(\mathcal{S}) \longrightarrow \textsc{PreAlg}_{\textsc{Setoid}}(\mathcal{S}).$$

This gives the required functors for the adjunction on the level of algebras, and the next step is to obtain the new unit and counit. The main idea is to show that the unit and counit are algebra homormophisms.

Proposition 6.4 *Given are categories $\mathcal{C}$ and $\mathcal{D}$, functors $A_1 : \mathcal{C} \longrightarrow \mathcal{C}$, $A_2 : \mathcal{D} \longrightarrow \mathcal{D}$, and $L : \mathcal{C} \longrightarrow \mathcal{D}$, and a natural transformation $n : A_2 \circ L \Longrightarrow L \circ A_1$. In addition, suppose we a functor $R : \mathcal{D} \longrightarrow \mathcal{C}$ together with a natural transformation $m : A_1 \circ R \Longrightarrow R \circ A_2$ and an adjunction $L \dashv R$ with unit η and counit ε. We also assume that for each $X : \mathcal{C}$ and $Y : \mathcal{D}$ the following diagrams commute*



Then the maps $\varepsilon\,X$ and $\eta\,Y$ are algebra homomorphisms for each $X : \mathcal{C}$ and $Y : \mathcal{D}$. Furthermore, we have an adjunction $L^{\textsc{PreAlg}} \dashv R^{\textsc{PreAlg}}$.

For the verification of the conditions of this proposition for the path setoid and quotient, we refer the reader to the formalization. Now we conclude

Lemma 6.5 *We have an adjunction $\mathcal{Q}^{\textsc{PreAlg}} \dashv \pi_0^{\textsc{PreAlg}}$.*

6.3 Algebras

If $\mathcal{C}$ is a category and P is a family of propositions on $\mathcal{C}$, then we write $\mathsf{full}(\mathcal{C}, P)$ for the full subcategory of $\mathcal{C}$ in which each object satisfies P. First, we look at a way to obtain adjunctions between full subcategories.

Lemma 6.6 *Let $\mathcal{C}$ and $\mathcal{D}$ be categories, let P_1 and P_2 be families of propositions on $\mathcal{C}$ and $\mathcal{D}$ respectively, and suppose we have a functor $F : \mathcal{C} \longrightarrow \mathcal{D}$. If for each object $x : \mathcal{C}$ we have $P_1\,x \to P_2(F\,x)$, then we get a functor $F_{\mathsf{sub}} : \mathsf{full}(\mathcal{C}, P_1) \longrightarrow \mathsf{full}(\mathcal{D}, P_2)$.*

Proposition 6.7 *Let $\mathcal{C}$ and $\mathcal{D}$ be categories, let P_1 and P_2 be families of propositions on $\mathcal{C}$ and $\mathcal{D}$ respectively, and suppose we have an adjunction $L \dashv R$ with $L : \mathcal{C} \longrightarrow \mathcal{D}$. If for each object $x : \mathcal{C}$ we have $P_1\,x \to P_2(L\,x)$ and for each $y : \mathcal{D}$ we have $P_2\,x \to P_1(R\,x)$, then we get an adjunction $L_{\mathsf{sub}} \dashv R_{\mathsf{sub}}$.*

Now we would like to apply this proposition to the adjunction obtained from Lemma 6.5. For that, we first need to calculate the action of the endpoints. This is done by the following lemma.

Lemma 6.8 *Given is an endpoint $e : \mathcal{E}_A(P, Q)$. For every set prealgebra X, we have the following equality*

$$\pi_0\,(\llbracket e \rrbracket\,X) \circ n_1^P\,X = n_1^Q\,X \circ \langle e \rangle\,(\pi_0^{\textsc{PreAlg}}\,X).$$

For every setoid prealgebra X, we have the following equality.

$$\mathcal{Q}\,(\langle e \rangle\,X) \circ n_2^P\,X = \llbracket e \rrbracket\,(\mathcal{Q}^{\textsc{PreAlg}}\,X).$$

For the quotient, a similar lemma is required and for its precise formulation, we refer the reader the formalization. With all this in place, we obtain the desired adjunction.

Theorem 6.9 *We have an adjunction $\mathcal{Q}^{\textsc{Alg}} \dashv \pi_0^{\textsc{Alg}}$.*

6.4 Initial Setoid Algebra

The initial setoid is constructed in two steps. First we define its carrier as the initial algebra on $\mathcal{S}_{\mathsf{pt}}$ and then we define the equivalence relation as the least congruence relation containing the equations in $\mathcal{S}$ and preserving equality of the point constructor.

Lemma 6.10 *The category $\textsc{PreAlg}_{\textsc{hSet}}(\mathcal{S}_{pt})$ has an initial object.*

This follows from Adámek's theorem and the fact that the functor $\llbracket P \rrbracket$ is ω-continuous for each P. The colimits required for this theorem are constructed from quotient types. We denote the initial object of this category by $(\mathcal{X}, f)$. The carrier of the desired setoid is $\mathcal{X}$ and the next step is to define the equivalence relation on $\mathcal{X}$. The main difficulty of defining this relation, is that we need to be able to lift f to a setoid morphism meaning that $\llbracket \mathcal{S}_{\mathsf{pt}} \rrbracket \mathcal{X}$ must be defined correctly. For that reason, we first define a relation on $\llbracket P \rrbracket\,\mathcal{X}$ for $P : \mathcal{P}$.

Definition 6.11 Given is $P : \mathcal{P}$ and a signature $\mathcal{S}$. The relation $\mathcal{R}^*\ P$ on $[\![P]\!]\ \mathcal{X}$ is inductively generated by the constructors

$$\frac{x : [\![P]\!]\ \mathcal{X}}{\mathbf{refl}\ x : \mathcal{R}^*\ P\ x\ x} \qquad \frac{r : \mathcal{R}^*\ P\ x\ y}{\mathbf{sym}\ r : \mathcal{R}^*\ P\ y\ x} \qquad \frac{r_1 : \mathcal{R}^*\ P\ x\ y \qquad r_2 : \mathcal{R}^*\ P\ y\ z}{\mathbf{trans}\ r_1\ r_2 : \mathcal{R}^*\ P\ x\ z}$$

$$\frac{r : \mathcal{R}^*\ P_1\ x\ y}{\mathbf{inl}\ r : \mathcal{R}^*\ (P_1 + P_2)\ (\mathsf{inl}\ x)\ (\mathsf{inl}\ y)} \qquad \frac{r : \mathcal{R}^*\ P_2\ x\ y}{\mathbf{inr}\ r : \mathcal{R}^*\ (P_1 + P_2)\ (\mathsf{inr}\ x)\ (\mathsf{inr}\ y)}$$

$$\frac{r_1 : \mathcal{R}^*\ P_1\ x_1\ x_2 \qquad r_2 : \mathcal{R}^*\ P_2\ y_1\ y_2}{\mathbf{pair}\ r_1\ r_2 : \mathcal{R}^*\ (P_1 \times P_2)\ (x_1, x_2)\ (y_1, y_2)}$$

$$\frac{j : \mathcal{S}_{\mathsf{pth}} \qquad x : [\![\mathcal{S}_{\mathsf{arg}}\ j]\!]\ \mathcal{X}}{\mathbf{path}\ j\ x : \mathcal{R}^*\ \mathbf{I}\ ([\![\mathcal{S}_{\mathsf{lhs}}\ j]\!]\ \mathcal{X}\ x)\ ([\![\mathcal{S}_{\mathsf{rhs}}\ j]\!]\ \mathcal{X}\ x)} \qquad \frac{r : \mathcal{R}^*\ \mathcal{S}_{\mathsf{pt}}\ x\ y}{\mathbf{cong}\ r : \mathcal{R}^*\ \mathbf{I}\ (f\ x)\ (f\ y)}$$

The extra argument P represents the sort of x, and we leave the signature $\mathcal{S}$ implicit when talking about this relation. Now we define a relation on $\mathcal{X}$ by $\mathcal{R}\ x\ y = \|\mathcal{R}^*\ \mathbf{I}\ x\ y\|$. Because of the truncation, each $\mathcal{R}\ x\ y$ is a proposition. Note that $\mathcal{R}$ is an equivalence relation, because of the constructor **refl**, **sym**, and **trans**. In addition, all equations in $\mathcal{S}$ are validated in $\mathcal{R}$ by **path**. Finally, with the constructors **pair**, **inl**, **inr**, and **cong**, we can show that f preserves $\mathcal{R}$. Hence, we obtain a setoid $I := (\mathcal{X}, \mathcal{R}^*)$ together with a setoid morphism $\varphi : \langle P \rangle\ I \to I$. To show that this is the initial setoid algebra, we first need a property of $\mathcal{R}$, which allows us to construct maps from I to algebras.

Lemma 6.12 *If g is a prealgebra morphism from $\mathcal{X}$ to an algebra, then g preserves $\mathcal{R}$.*

Now we show that I is an initial object in $\textsc{Alg}_{\textsc{hSet}}(\mathcal{S})$. To show that we always have a homomorphism from I to an algebra X, we use that $\mathcal{X}$ is the initial prealgebra and Lemma 6.12. Furthermore, note that two setoid morphisms are equal if their underlying carriers are equal and two algebra morphisms are also equal if their underlying carrier is equal. Hence, the uniqueness also follows from initiality of $\mathcal{R}^*$. All in all, we get

Theorem 6.13 *The algebra I is the initial object of $\textsc{Alg}_{\textsc{Setoid}}(\mathcal{S})$.*

From Theorems 6.9 and 6.13 we conclude

Corollary 6.14 *For each signature $\mathcal{S}$, there is a HIT on $\mathcal{S}$.*

7 Consequences

7.1 Uniqueness of HITs

Now we discuss two consequences of this construction. A first property of higher inductive types, is that they are unique up to equality. For that, we first show that higher inductives are initial algebras. This requires the well-known result that induction implies initiality [11,39]. Note that in Section 5 we used the converse statement, namely that initiality implies induction.

To show a HIT H is initial, we must show that for each algebra X the set of morphisms from H to X is contractible. This means that there is precisely one homomorphism from H to X. To show the existence, we use the following two lemmata.

Lemma 7.1 *Let X and Y be algebras on $\mathcal{S}$. Then there is a displayed algebra $\mathsf{const}\ Y$ on X of which the underlying type family is Y on each point of X.*

Lemma 7.2 *Let X be Y be algebras on $\mathcal{S}$. If we have a displayed algebra map from X to $\mathsf{const}\ Y$, then we have an algebra homomorphism from X to Y.*

Corollary 7.3 *If X is a HIT for $\mathcal{S}$, then for each algebra Y we have an algebra map from X to Y.*

To show uniqueness, we use an alternative induction principle for families of propositions. More specifically, we use the following lemma.

Lemma 7.4 *Let H be a HIT of $\mathcal{S}$ and suppose Y is a family of propositions on H. If we have an operation*

$$c : \prod(z : [\![\mathcal{S}_{pt}]\!]H), \overline{\mathcal{S}_{pt}}\ Y\ z \to Y\ (\mathsf{point}_X\ z),$$

then we have a map $\prod(x : H), Y\ x$.

To prove the uniqueness of the map, we use function extensionality and the previous lemma. This is sufficient to conclude that higher inductive types are initial objects.

Proposition 7.5 *If H is a HIT for $\mathcal{S}$, then H is an initial $\mathcal{S}$-algebra.*

Now we take advantage of Proposition 4.4 where we proved that the category of algebras is univalent. Since initial objects in a univalent category are unique up to equality, we can immediately conclude that HITs are actually unique up to equality. All in all, we get

Corollary 7.6 *If H_1 and H_2 are HITs for $\mathcal{S}$, then the underlying $\mathcal{S}$-algebras of H_1 and H_2 are equal.*

7.2 Path Spaces of HITs

The construction of HITs also allows us to characterize the path space up to equivalence. For this, we first recall the characterization of the path space of the quotient type.

Proposition 7.7 (Lemma 10.1.8 from [41]) *Let X be a type and let R be an equivalence relation on X. Then for each $x, y : X$, we have an equivalence $R\,x\,y \simeq \mathbf{class}\,x = \mathbf{class}\,y$.*

Now we specialize this theorem to HITs constructed according to Theorem 6.14. This allows us to conclude that the path space of set truncated HITs is freely generated.

Corollary 7.8 *Let $\mathcal{S}$ be a signature. Then for each $x, y : \mathcal{X}$ we have an equivalence $\mathcal{R}\,x\,y \simeq \mathbf{class}\,x = \mathbf{class}\,y$ using $\mathcal{X}$ and $\mathcal{R}$ as defined in the previous section.*

Hence, since HITs are unique, they must be equal to the HIT we constructed in the previous section, and this determines the path spaces of HITs. Note that this corollary gives an induction principle for the type $\mathbf{class}\,x = \mathbf{class}\,y$, because $\mathcal{R}$ is defined as the propositional truncation of an inductive type, For this reason, we conclude that the path space of higher inductive types are freely generated.

8 Conclusion and Further Work

We have shown how to construct finitary set-truncated higher inductive types using the quotient and propositional truncation in Theorem 6.14. Since truncations and the quotient can be constructed from non-recursive HITs, this proves that all finitary set-truncated HITs can be constructed from non-recursive ones. Besides that, we took advantage of this construction to show uniqueness of HITs in Corollary 7.6 and to characterize their path space in Corollary 7.8.

The method used in this paper crucially relies on the polynomials being finitary. To show that Infinitary polynomials commute with quotients, one needs the axiom of choice [16]. Furthermore, Lumsdaine and Shulman gave an infinitary HIT which cannot be constructed from pushouts, and thus this result cannot be extended to infinitary polynomials.

However, a possible extension would be by allowing the arguments of point constructors to depend on previous ones [4]. For a construction of all finitary HITs, higher path constructors need to be allowed and the HITs need to be interpreted in some higher category. A first step towards that goal, would be adapting this construction to bicategory theory [1,14] and constructing higher inductive 1-types using Dybjer's and Moeneclaey's interpretation of those in groupoids [20]. For non-truncated types, a type theoretic definition of ω-groupoid would be required [8,14,21,24]. In both cases, one also needs to interpret the endpoints of homotopies. In addition, the path computation rules become relevant. Proving the induction principle from initiality is more difficult then, because equality becomes proof relevant giving nontrivial transports. In this setting, Corollary 7.8 is more interesting, because the computation rules of path constructors become nontrivial.

References

[1] Benedikt Ahrens, Dan Frumin, Marco Maggesi, and Niels van der Weide. Bicategories in Univalent Foundations. *arXiv preprint arXiv:1903.01152*, 2019.

[2] Benedikt Ahrens, Krzysztof Kapulkin, and Michael Shulman. Univalent Categories and the Rezk Completion. *Mathematical Structures in Computer Science*, 25(5):1010–1039, 2015.

[3] Benedikt Ahrens and Peter LeFanu Lumsdaine. Displayed Categories. *Logical Methods in Computer Science*, 15(1), 2019.

[4] Thorsten Altenkirch, Paolo Capriotti, Gabe Dijkstra, Nicolai Kraus, and Fredrik Nordvall Forsberg. Quotient Inductive-Inductive Types. In *Foundations of Software Science and Computation Structures - 21st International Conference, FOSSACS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings*, pages 293–310, 2018.

[5] Thorsten Altenkirch, Nils Anders Danielsson, and Nicolai Kraus. Partiality, Revisited - The Partiality Monad as a Quotient Inductive-Inductive Type. In *Foundations of Software Science and Computation Structures - 20th International Conference, FOSSACS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*, pages 534–549, 2017.

[6] Thorsten Altenkirch and Ambrus Kaposi. Type Theory in Type Theory using Quotient Inductive Types. In *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, pages 18–29, 2016.

[7] Thorsten Altenkirch and Ambrus Kaposi. Normalisation by Evaluation for Type Theory, in Type Theory. *Logical Methods in Computer Science*, 13(4), 2017.

[8] Thorsten Altenkirch and Ondrej Rypacek. A Syntactical Approach to Weak ω-Groupoids. In *Computer Science Logic (CSL'12) - 26th International Workshop/21st Annual Conference of the EACSL, CSL 2012, September 3-6, 2012, Fontainebleau, France*, pages 16–30, 2012.

[9] Carlo Angiuli, Robert Harper, and Todd Wilson. Computational Higher-Dimensional Type Theory. In *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL 2017, Paris, France, January 18-20, 2017*, pages 680–693, 2017.

[10] Carlo Angiuli, Edward Morehouse, Daniel R. Licata, and Robert Harper. Homotopical Patch Theory. *J. Funct. Program.*, 26:e18, 2016.

[11] Steven Awodey, Nicola Gambino, and Kristina Sojakova. Inductive Types in Homotopy Type Theory. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*, pages 95–104, 2012.

[12] Henning Basold, Herman Geuvers, and Niels van der Weide. Higher Inductive Types in Programming. *J. UCS*, 23(1):63–88, 2017.

[13] Marc Bezem, Thierry Coquand, and Simon Huber. A Model of Type Theory in Cubical Sets. In *19th International Conference on Types for Proofs and Programs, TYPES 2013, April 22-26, 2013, Toulouse, France*, pages 107–128, 2013.

[14] Paolo Capriotti and Nicolai Kraus. Univalent Higher Categories via Complete Semi-Segal Types. *Proceedings of the ACM on Programming Languages*, 2(POPL):44, 2017.

[15] Evan Cavallo and Robert Harper. Higher Inductive Types in Cubical Computational Type Theory. *PACMPL*, 3(POPL):1:1–1:27, 2019.

[16] James Chapman, Tarmo Uustalu, and Niccolò Veltri. Quotienting the Delay Monad by Weak Bisimilarity. *Mathematical Structures in Computer Science*, 29(1):67–92, 2019.

[17] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. Cubical Type Theory: a Constructive Interpretation of the Univalence Axiom. *CoRR*, abs/1611.02108, 2016.

[18] Thierry Coquand, Simon Huber, and Anders Mörtberg. On Higher Inductive Types in Cubical Type Theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09-12, 2018*, pages 255–264, 2018.

[19] Peter Dybjer. Inductive Families. *Formal aspects of computing*, 6(4):440–465, 1994.

[20] Peter Dybjer and Hugo Moeneclaey. Finitary Higher Inductive Types in the Groupoid Model. *Electr. Notes Theor. Comput. Sci.*, 336:119–134, 2018.

[21] Eric Finster and Samuel Mimram. A Type-Theoretical Definition of Weak ω-Categories. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*, pages 1–12, 2017.

[22] Dan Frumin, Herman Geuvers, Léon Gondelman, and Niels van der Weide. Finite Sets in Homotopy Type Theory. In *Proceedings of the 7th ACM SIGPLAN International Conference on Certified Programs and Proofs, CPP 2018, Los Angeles, CA, USA, January 8-9, 2018*, pages 201–214, 2018.

[23] Claudio Hermida and Bart Jacobs. Structural Induction and Coinduction in a Fibrational Setting. *Information and computation*, 145(2):107–152, 1998.

[24] André Hirschowitz, Tom Hirschowitz, and Nicolas Tabareau. Wild ω-Categories for the Homotopy Hypothesis in Type Theory. In *13th International Conference on Typed Lambda Calculi and Applications, TLCA 2015, July 1-3, 2015, Warsaw, Poland*, pages 226–240, 2015.

[25] Martin Hofmann and Thomas Streicher. The Groupoid Model Refutes Uniqueness of Identity Proofs. In *Proceedings of the Ninth Annual Symposium on Logic in Computer Science (LICS '94), Paris, France, July 4-7, 1994*, pages 208–212, 1994.

[26] Ambrus Kaposi and András Kovács. A Syntax for Higher Inductive-Inductive Types. In *3rd International Conference on Formal Structures for Computation and Deduction, FSCD 2018, July 9-12, 2018, Oxford, UK*, pages 20:1–20:18, 2018.

[27] Ambrus Kaposi, András Kovács, and Thorsten Altenkirch. Constructing Quotient Inductive-Inductive Types. *PACMPL*, 3(POPL):2:1–2:24, 2019.

[28] Chris Kapulkin and Peter LeFanu Lumsdaine. The Simplicial Model of Univalent Foundations (after Voevodsky), 2012.

[29] Nicolai Kraus. Constructions with Non-Recursive Higher Inductive Types. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 595–604, 2016.

[30] Nicolai Kraus and Jakob von Raumer. Path Spaces of Higher Inductive Types in Homotopy Type Theory. *CoRR*, abs/1901.06022, 2019.

[31] Daniel R. Licata and Eric Finster. Eilenberg-MacLane Spaces in Homotopy Type Theory. In *Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS), CSL-LICS '14, Vienna, Austria, July 14 - 18, 2014*, pages 66:1–66:9, 2014.

[32] Daniel R. Licata and Michael Shulman. Calculating the Fundamental Group of the Circle in Homotopy Type Theory. In *28th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2013, New Orleans, LA, USA, June 25-28, 2013*, pages 223–232, 2013.

[33] Peter LeFanu Lumsdaine. Weak ω-categories from Intensional Type Theory. In *International Conference on Typed Lambda Calculi and Applications*, pages 172–187. Springer, 2009.

[34] Peter LeFanu Lumsdaine and Mike Shulman. Semantics of Higher Inductive Types. *arXiv preprint arXiv:1705.07088*, 2017.

[35] Saunders Mac Lane. *Categories for the Working Mathematician*, volume 5. Springer Science & Business Media, 2013.

[36] Hugo Moeneclaey. A Schema for Higher Inductive Types of Level One and Its Interpretation. *Internship report, supervised by Peter Dybjer, ENS Paris-Saclay*, 2016.

[37] Egbert Rijke. The Join Construction. *arXiv preprint arXiv:1701.07538*, 2017.

[38] Egbert Rijke and Bas Spitters. Sets in Homotopy Type Theory. *Mathematical Structures in Computer Science*, 25(5):1172–1202, 2015.

[39] Kristina Sojakova. Higher Inductive Types as Homotopy-Initial Algebras. In *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, pages 31–42, 2015.

[40] Ross Street. The Formal Theory of Monads. *Journal of Pure and Applied Algebra*, 2(2):149–168, 1972.

[41] The Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. https://homotopytypetheory.org/book, Institute for Advanced Study, 2013.

[42] Benno Van Den Berg and Richard Garner. Types are Weak ω-Groupoids. *Proceedings of the London Mathematical Society*, 102(2):370–394, 2011.

[43] Floris van Doorn. Constructing the Propositional Truncation using Non-Recursive HITs. In *Proceedings of the 5th ACM SIGPLAN Conference on Certified Programs and Proofs, Saint Petersburg, FL, USA, January 20-22, 2016*, pages 122–129, 2016.

[44] Vladimir Voevodsky, Benedikt Ahrens, Daniel Grayson, et al. UniMath — a computer-checked library of univalent mathematics. available at https://github.com/UniMath/UniMath.