Constructing Higher Inductive Types as Groupoid Quotients

Niels van der Weide
Institute for Computation and Information Sciences
Radboud Universiteit
Nijmegen, The Netherlands
nweide@cs.ru.nl

Abstract

In this paper, we show that all finitary 1-truncated higher inductive types (HITs) can be constructed from the groupoid quotient. We start by defining internally a notion of signatures for HITs, and for each signature, we construct a bicategory of algebras in 1-types and in groupoids. We continue by proving initial algebra semantics for our signatures. After that, we show that the groupoid quotient induces a biadjunction between the bicategories of algebras in 1-types and in groupoids. We finish by constructing a biinitial object in the bicategory of algebras in groupoids. From all this, we conclude that all finitary 1-truncated HITs can be constructed from the groupoid quotient. All the results are formalized over the UniMath library of univalent mathematics in Coq.

CCS Concepts: • **Theory of computation** → **Type theory**; **Constructive mathematics**.

Keywords: higher inductive types, homotopy type theory, Coq, bicategories

ACM Reference Format:
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 8–11, 2020, Saarbrücken, Germany*. ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3373718.3394803

1 Introduction

The Martin-Löf identity type, also known as *propositional equality*, represents provable equality in type theory [52]. This type is defined polymorphically over all types and has a single introduction rule representing reflexivity. The eliminator, often called the J-rule or path induction, is used to prove symmetry and transitivity. Note that in particular, we can talk about the identity type of an already established identity type. This can be iterated to obtain an infinite tower of types, which has the structure of an ∞-groupoid [50, 60].

The J-rule is also the starting point of *homotopy type theory* [59]. In that setting, types are seen as spaces, inhabitants are seen as points, proofs of identity are seen as paths, and paths between paths are seen as homotopies. In mathematical terms, type theory can be interpreted in many Quillen model categories [15], as for example simplicial sets [41]. In the resulting model, not every inhabitant of the identity type is equal to reflexivity, which is also the case in the groupoid model [37, 38] and the cubical sets model [18].

If we assume enough axioms, then we can construct types for which we can prove that not every two inhabitants of the identity type are equal. One example is the universe if one assumes the univalence axiom [59]. Other examples can be obtained by using *higher inductive types* (HITs).

Higher inductive types generalize inductive types by allowing constructors for paths, paths between paths, and so on. While inductive types are specified by giving the arities of the operations [27], for higher inductive types one must also specify the arities of the paths, paths between paths, and so on. The resulting higher inductive type is freely generated by the provided constructors. To make this concrete, let us look at some examples [59]:

$$\begin{aligned}&\texttt{Inductive } S^1 :=\\ &\mid \mathbf{base}_{S^1} : S^1\\ &\mid \mathbf{loop}_{S^1} : \mathbf{base}_{S^1} = \mathbf{base}_{S^1}\end{aligned}$$

$$\begin{aligned}&\texttt{Inductive } \mathcal{T}^2 :=\\ &\mid \mathbf{base} : \mathcal{T}^2\\ &\mid \mathbf{loop_l}, \mathbf{loop_r} : \mathbf{base} = \mathbf{base}\\ &\mid \mathbf{surf} : \mathbf{loop_l} \bullet \mathbf{loop_r} = \mathbf{loop_r} \bullet \mathbf{loop_l}\end{aligned}$$

The first one, S^1, represents the circle. It is generated by a point constructor $\mathbf{base}_{S^1} : S^1$ and a path constructor $\mathbf{loop}_{S^1} : \mathbf{base}_{S^1} = \mathbf{base}_{S^1}$. The second one, $\mathcal{T}^2$, represents the torus. This type is generated by a point constructor **base**, two path constructors $\mathbf{loop_l}$ and $\mathbf{loop_r}$ of type $\mathbf{base} = \mathbf{base}$, and a homotopy constructor $\mathbf{surf} : \mathbf{loop_l} \bullet \mathbf{loop_r} = \mathbf{loop_r} \bullet \mathbf{loop_l}$ where $p \bullet q$ denotes the concatenation of p and q. Note that constructors depend on previously given constructors in the specification. For both types, introduction, elimination, and computation rules can be given [59].

LICS '20, July 8–11, 2020, Saarbrücken, Germany

ACM ISBN 978-1-4503-7104-9/20/07.
https://doi.org/10.1145/3373718.3394803

In this paper, we study a schema of higher inductive types that allows defining types by giving constructors for the points, paths, and homotopies. All of these constructors can be recursive, but they can only have a finite number of recursive arguments. Concretely, this means that every inhabitant can be constructed as a finitely branching tree. Note that recursion is necessary to cover examples such as the set truncation, algebraic theories, and the integers. Such a HIT is called *finitary*. A similar scheme was studied by Dybjer and Moeneclaey and they interpret HITs on this scheme in the groupoid model [28].

Say that a type X is *1-truncated* if for all $x, y : X$, $p, q : x = y$, and $r, s : p = q$ we have $r = s$, and a *1-type* is a type which is 1-truncated. In terms of the ∞-groupoid structure mentioned before, such types are 1-groupoids. An example of a 1-type is S^1 [49], which we mentioned before, and another one is the classifying space of a group [48]. Groupoids are related to 1-types via the *groupoid quotient* [58], which takes a groupoid G and returns a 1-type whose points are objects of G up to equivalence. Note that the types of univalent groupoids and of 1-types are equivalent [3].

The goal of this paper is to show that finitary 1-truncated higher inductive types can be derived from simpler principles. More specifically, every finitary 1-truncated HIT can be constructed in a type theory with propositional truncations, set quotients, and groupoid quotients. Note that the set quotient is a special instance of the groupoid quotient. The result of this paper can be used to simplify the semantic study of finitary 1-truncated HITs. Instead of verifying the existence of a wide class of HITs, one only needs to check the existence of propositional truncations and groupoid quotients.

The contributions of this paper are summarized as follows

- An internal definition of signatures for HITs which allows path and homotopy constructors (Definition 3.4);
- Bicategories of algebras in both 1-types and groupoids (Definition 3.16);
- A proof that biinitial algebras in 1-types satisfy the induction principle (Proposition 4.14);
- A biadjunction between the bicategories of algebras in 1-types and algebras in groupoids (Construction 5.12);
- A construction of 1-truncated HITs from the groupoid quotient (Construction 6.4), which shows that such HITs exist. This is the main contribution of this paper.

Related Work. Various schemes of higher inductive types have been defined and studied. Awodey *et al.* study inductive types in homotopy type theory and prove initial algebra semantics [14]. Sojakova extended their result to various higher inductive types, among which are the groupoid quotient, W-suspensions, and the torus [57, 58]. Basold *et al.* define a scheme for HITs allowing for both point and path constructors, but no higher constructors [16], and a similar scheme is given by Moeneclaey [54]. Dybjer and Moeneclaey extended this scheme by allowing homotopy constructors and they give semantics in the groupoid model [28]. In the framework of computational higher-dimensional type theory [11], Cavallo and Harper defined indexed cubical inductive types and prove canonicity [22]. Altenkirch *et al.* define quotient inductive-inductive types, which combine the features of quotient types with inductive-inductive types [5, 31]. Kovács and Kaposi extended this syntax to higher inductive-inductive types [39], which can be used to define not necessarily set-truncated types. The scheme studied in this paper, is most similar to the one by Dybjer and Moeneclaey [28] with the restriction that each type has a constructor indicating that the type is 1-truncated. In particular, this means that inductive-inductive types are not considered. Note that the HITs we study only have the right elimination property with respect to 1-types unlike W-suspensions [57, 58].

Higher inductive types have already been used for numerous applications. One of them is synthetic homotopy theory. Spaces, such as the real projective spaces, higher spheres, and Eilenberg-MacLane spaces, can be defined as higher inductive types [20, 46, 48, 59]. The resulting definitions are strong enough to determine homotopy groups [46, 49]. In addition, algebraic theories can be modeled as HITs, which allows one to define finite sets as a higher inductive type [33]. Other applications of HITs include homotopical patch theory, which provides a way to model version control systems [12], and modeling data types such as the integers [10, 16]. Besides, quotient inductive-inductive types can be used to define the partiality monad [6]. These types can also be used to define type theory within type theory [7] and to prove normalization [8]. Since the HITs in this paper are 1-truncated, they can capture algebraic theories while for examples such as real projective spaces and higher spheres, we can only define their 1-truncation.

Several classes of higher inductive types have already been reduced to simpler ones. Both Van Doorn and Kraus constructed propositional truncations from non-recursive higher inductive types [42, 62]. Using the join construction, Rijke constructed several examples of HITs, namely n-truncations, the Rezk completion, and set quotients [56]. Awodey *et al.* give an impredicative construction of finitary inductive types and some HITs [13]. Constructions of more general classes of HITs have also been given. Assuming UIP, Kaposi *et al.* constructed all finitary quotient inductive-inductive types from a single one [40], and without UIP, Van der Weide and Geuvers constructed all finitary set truncated HITs from quotients [61]. Note that these two works only concern set truncated HITs while our work concerns 1-truncated HITs. Furthermore, the HITs considered by Van der Weide and Geuvers are a special case of the HITs in this paper.

Lastly, an alternative way to verify the existence of higher inductive types, is by constructing them directly in a model. Coquand *et al.* interpreted several HITs in the cubical sets

model [18, 25]. Note that one can constructively prove univalence in the cubical sets model [24] and that cubical type theory satisfies homotopy canonicity [26]. Furthermore, cubical type theory has been implemented in Agda with support for higher inductive types [63]. Lumsdaine and Shulman give a semantical scheme for HITs and show that these can be interpreted in sufficiently nice model categories [51].

Formalization. All results in this paper are formalized in Coq [53] using UniMath [64]. The formalization uses the version with `git` hash 2dadfb61 and can be found here:

https://github.com/nmvdw/GrpdHITs/tree/LICS

Overview. We start by recalling the groupoid quotient and displayed bicategories in Section 2. Displayed bicategories are our main tool to construct the bicategory on algebras on a signature. In Section 3, we define signatures and show that each signature gives rise to a bicategory of algebras in both 1-types and groupoids. The notion of a higher inductive type on a signature is given in Section 4. There, we also prove initial algebra semantics, which says that biinitiality is a sufficient condition for being a HIT. To construct the desired higher inductive type, we use the groupoid quotient, and in Section 5 we lift this to a biadjunction on the level of algebras. As a consequence, constructing the initial algebra of a signature in groupoids is sufficient to construct the desired higher inductive type. In Section 6, we construct the desired initial algebra and we conclude that each signature has a higher inductive type. Lastly, we conclude in Section 7.

Notation. Let us recall some notation from HoTT which we use throughout this paper. The identity path is denoted by $\mathbf{idpath}(x)$ and the concatenation of paths $p : x = y$ and $q : y = z$ is denoted by $p \bullet q$. Given a type X with points $x, y : X$ and paths $p, q : x = y$, we call a path $s : p = q$ a *2-path*. A *proposition* is a type of which all inhabitants are equal. A *set* is a type X such that for all $x, y : X$ the type $x = y$ is a proposition. A *homotopy* between $f, g : X \to Y$ consists of a path $f(x) = g(x)$ for each $x : A$.

2 Preliminaries

2.1 Groupoid Quotient

Let us start by formally introducing the groupoid quotient [58]. The groupoid quotient is a higher dimensional version of the set quotient, so let us quickly recall the set quotient. Given a setoid (X, R) (a set X with an equivalence relation R valued in propositions on X), the set quotient gives a type X/R, which is X with the points identified according to R. Note that X/R always is a set since R becomes the equality.

Instead of a setoid, the groupoid quotient takes a groupoid as input. Recall that a groupoid is a category in which every morphism is invertible. In particular, each groupoid has identity morphisms, denoted by $\mathrm{id}(x)$, and a composition operation. The composition of f and g is denoted by $f \cdot g$. In addition, the type of morphisms from x to y is required to be a set. We write Grpd for the type of groupoids.

Given G : Grpd, the groupoid quotient gives a 1-type $\mathrm{GQuot}(G)$. In this type, the points are objects of G and these are identified according to the morphisms in G. In addition, the groupoid structure must be preserved. Informally, we define the groupoid quotient as the following HIT.

```
Inductive GQuot (G : Grpd) :=
| gcl : G → GQuot(G)
| gcleq : ∏(x, y : G)(f : G(x, y)), gcl(x) = gcl(y)
| ge : ∏(x : G), gcleq(id(x)) = idpath(gcl(x))
| gconcat : ∏(x, y, z : G)(f : G(x, y))(g : G(y, z)),
            gcleq(f · g) = gcleq(f) • gcleq(g)
| gtrunc : ∏(x, y : GQuot(G))(p, q : x = y)(r, s : p = q),
            r = s
```

To formally add this type to our theory, we need to provide introduction, elimination, and computation rules for $\mathrm{GQuot}(G)$. Formulating the elimination principle requires two preliminary notions. These are inspired by the work by Licata and Brunerie [47]. The first of these gives paths in a dependent type over a path in the base.

Definition 2.1. Given a type X : Type, a type family $Y : X \to$ Type, points $x_1, x_2 : X$, a path $p : x_1 = x_2$, and points $\overline{x_1} : Y(x_1)$ and $\overline{x_2} : Y(x_2)$ over x_1 and x_2 respectively, we define the type $\overline{x_1} =^Y_p \overline{x_2}$ of **paths over** p from $\overline{x_1}$ to $\overline{x_2}$ by path induction on p by saying that the paths over the identity path $\mathbf{idpath}(x)$ from $\overline{x_1}$ to $\overline{x_2}$ are just paths $\overline{x_1} = \overline{x_2}$.

Note that the groupoid quotient also has constructors for paths between paths. This means that we also need a dependent version of 2-paths, and inspired by the terminology of globular sets, we call these *globes* over a given 2-path. We define them as follows.

Definition 2.2. Let X, Y, and x_1, x_2 be as in Definition 2.1. Suppose, that we paths $p, q : x_1 = x_2$, a 2-path $g : p = q$, and paths $\overline{p} : \overline{x_1} =_p \overline{x_2}$ and $\overline{q} : \overline{x_1} =_q \overline{x_2}$ over p and q respectively, we define the type $\overline{p} =_g \overline{q}$ of **globes over** g from $\overline{p}$ to $\overline{q}$ by path induction on g by saying that the paths over the identity path $\mathbf{idpath}(p)$ are just paths $\overline{p} = \overline{q}$.

From this point on, we assume that our type theory has the groupoid quotient. More specifically, we assume the following axiom.

Axiom 2.3. For each groupoid G there is a type $\mathrm{GQuot}(G)$ which satisfies the rules in Figure 1.

Note that there are no computation rules for **gconcat**, **ge**, and **gtrunc**, because such rules follow from the fact that Y is a family of 1-types.

2.2 Bicategory Theory

The upcoming construction makes heavy use of notions from bicategory theory [17, 45] and in particular, the displayed machinery introduced by Ahrens *et al.* [2]. Here we recall some examples of bicategories and the basics of displayed bicategories.

Introduction rules:

$$\frac{x : G}{\mathbf{gcl}(x) : \mathrm{GQuot}(G)} \qquad \frac{x, y : G \qquad f : G(x, y)}{\mathbf{gcleq}(f) : \mathbf{gcl}(x) = \mathbf{gcl}(y)} \qquad \frac{x : G}{\mathbf{ge}(x) : \mathbf{gcleq}(\mathrm{id}(x)) = \mathbf{idpath}(\mathbf{gcl}(x))}$$

$$\frac{x, y, z : G \qquad f : G(x, y) \qquad g : G(y, z)}{\mathbf{gconcat}(f, g) : \mathbf{gcleq}(f \cdot g) = \mathbf{gcleq}(f) \bullet \mathbf{gcleq}(g)} \qquad \frac{x, y : \mathrm{GQuot}(G) \qquad p, q : x = y \qquad r, s : p = q}{\mathbf{gtrunc}(r, s) : r = s}$$

Elimination rule:

$$\frac{\begin{array}{c} Y : \mathrm{GQuot}(G) \to \text{1-Type} \qquad \mathrm{gcl}_Y : \prod(x : G), Y(\mathbf{gcl}(x)) \\ \mathrm{gcleq}_Y : \prod(x, y : G)(f : G(x, y)), \mathrm{gcl}_Y(x) =_{\mathbf{gcleq}(f)} \mathrm{gcl}_Y(y) \\ \mathrm{ge}_Y : \prod(x : G), \mathrm{gcleq}_Y(\mathrm{id}(x)) =_{\mathbf{ge}(x)} \mathbf{idpath}(\mathrm{gcl}_Y(x)) \\ \mathrm{gconcat}_Y : \prod(x, y, z : G)(f : G(x, y))(g : G(y, z)), \mathrm{gcleq}_Y(f \cdot g) =_{\mathbf{gconcat}(f,g)} \mathrm{gcleq}_Y(f) \bullet \mathrm{gcleq}_Y(g) \end{array}}{\mathbf{gind}(\mathrm{gcl}_Y, \mathrm{gcleq}_Y, \mathrm{ge}_Y, \mathrm{gconcat}_Y) : \prod(x : \mathrm{GQuot}(G)), Y(x)}$$

Computation rules:

For **gcl**: $\mathbf{gind}(\mathrm{gcl}_Y, \mathrm{gcleq}_Y, \mathrm{ge}_Y, \mathrm{gconcat}_Y)(\mathbf{gcl}(x)) \equiv \mathrm{gcl}_Y(x)$

For **gcleq**: $\mathrm{apd}\ (\mathbf{gind}(\mathrm{gcl}_Y, \mathrm{gcleq}_Y, \mathrm{ge}_Y, \mathrm{gconcat}_Y))\ (\mathbf{gcleq}(f)) = \mathrm{gcleq}_Y(f)$

Figure 1. Introduction, elimination, and computation rules for the groupoid quotient [58].

Recall that a bicategory consists of objects, 1-cells between objects, and 2-cells between 1-cells. The type of 1-cells from x to y is denoted by $x \to y$ and the type of 2-cells from f to g is denoted by $f \Rightarrow g$. Note that the type $f \Rightarrow g$ is required to be a set. There are identity 1-cells and 2-cells denoted by id_1 and id_2 respectively, composition of 1-cells f and g is denoted by $f \cdot g$, and the vertical composition of 2-cells θ and θ' is denoted by $\theta \bullet \theta'$. The left whiskering of a 2-cell θ with 1-cell f is denoted by $f \lhd \theta$ and right whiskering of θ with a 1-cell g is denoted by $\theta \rhd g$. Unitality and associativity of vertical composition of 2-cells hold strictly while for 1-cells, these laws only hold up to an invertible 2-cell.

Let us fix some notation before continuing. Given bicategories B_1 and B_2, we write $\mathrm{Pseudo}(\mathsf{B}_1, \mathsf{B}_2)$ for the type of pseudofunctors from B_1 to B_2. The type of pseudotransformations from F to G is denoted by $F \Rightarrow G$ and the type of modifications from θ to θ' is denoted by $\theta \Rrightarrow \theta'$ [45]. Lastly, the type of biadjunctions between B_1 and B_2 is denoted by $L \dashv R$ where $L : \mathrm{Pseudo}(\mathsf{B}_1, \mathsf{B}_2)$ and $R : \mathrm{Pseudo}(\mathsf{B}_2, \mathsf{B}_1)$. If we have $L \dashv R$, we say that L is *left biadjoint* to R. The definition of biadjoint we use, is similar to the one used by Gurski [34], but without any coherencies. Beside these standard notions, we use two bicategories: 1-Type and Grpd.

Example 2.4. We have

- a bicategory 1-Type whose objects are 1-types, 1-cells are functions, and 2-cells are homotopies;
- a bicategory Grpd of groupoids whose objects are (not necessarily univalent) groupoids, 1-cells are functors, and 2-cells are natural transformations.

Next we discuss *displayed bicategories*, which is our main tool to define bicategories of algebras on a signature. Intuitively, a displayed bicategory D over B represents structure and properties to be added to B. Displayed bicategories generalize displayed categories to the bicategorical setting [4]. Each such D gives rise to a total bicategory $\int \mathsf{D}$. The full definition can be found in the paper by Ahrens *et al.* [2], and here, we only show a part.

Definition 2.5. Let B be a bicategory. A **displayed bicategory** D over B consists of

- For each $x : \mathsf{B}$ a type $\mathsf{D}(x)$ of **objects over** x;
- For each $f : x \to y$, $\overline{x} : \mathsf{D}(x)$ and $\overline{y} : \mathsf{D}(y)$, a type $\overline{x} \xrightarrow{f} \overline{y}$ of **1-cells over** f;
- For each $\theta : f \Rightarrow g$, $\overline{f} : \overline{x} \xrightarrow{f} \overline{y}$, and $\overline{g} : \overline{x} \xrightarrow{g} \overline{y}$, a *set* $\overline{f} \overset{\theta}{\Rightarrow} \overline{g}$ of **2-cells over** θ.

In addition, there are identity cells and there are composition and whiskering operations. The composition of displayed 1-cells f and g is denoted by $f \cdot g$, the displayed identity 1-cell is denoted by $\mathrm{id}_1(x)$. The vertical composition of 2-cells θ and θ' is denoted by $\theta \bullet \theta'$, the left and right whiskering is denoted by $f \lhd \theta$ and $\theta \rhd f$ respectively, and the identity 2-cell is denoted by $\mathrm{id}_2(f)$.

Definition 2.6. Let B be a bicategory and let D be a displayed bicategory over B. We define the **total bicategory** $\int D$ as the bicategory whose objects of $\int D$ are just dependent pairs $(x, \overline{x})$ with x in B and $\overline{x}$ in $\mathsf{D}(x)$. The 1-cells and 2-cells in $\int D$ are defined similarly. In addition, we define the **projection** $\pi_D : \mathrm{Pseudo}(\int D, \mathsf{B})$ to be the pseudofunctor which takes the first component of each pair.

Let us finish this section by defining the displayed bicategories we need in the remainder of this paper. Motivation and explanation of Examples 2.7 and 2.9 is given by Ahrens *et al.* [2].

Example 2.7. Given a bicategory B and a pseudofunctor $F : \mathsf{Pseudo}(\mathsf{B}, \mathsf{B})$, we define a displayed bicategory $\mathsf{DFalg}(F)$ over B such that

- the objects over $x : \mathsf{B}$ are 1-cells $h_x : F(x) \to x$;
- the 1-cells over $f : x \to y$ from h_x to h_y are invertible 2-cells $\tau_f : h_x \cdot f \Rightarrow F(f) \cdot h_y$;
- the 2-cells over $\theta : f \Rightarrow g$ from τ_f to τ_g are equalities

$$h_x \lhd \theta \bullet \tau_g = \tau_f \bullet F(\theta) \rhd h_y.$$

Example 2.8. Given a bicategory B, a type I, and for each $i : I$ a displayed bicategory D_i over B, we define a displayed bicategory $\prod(i : I), \mathsf{D}_i$ over B such that

- the objects over $x : \mathsf{B}$ are functions $\prod(i : I), \mathsf{D}_i(x)$;
- the 1-cells over $f : x \to y$ from $\overline{x}$ to $\overline{y}$ are functions $\prod(i : I), \overline{x}(i) \xrightarrow{f} \overline{y}(i)$;
- the 2-cells over $\theta : f \Rightarrow g$ from $\overline{f}$ to $\overline{g}$ are functions $\prod(i : I), \overline{f}(i) \overset{\theta}{\Rightarrow} \overline{g}(i)$.

Example 2.9. Let B be a bicategory with a displayed bicategory D over it. Now suppose that we have pseudofunctors $S, T : \mathsf{Pseudo}(\mathsf{B}, \mathsf{B})$ and two pseudotransformations $l, r : \pi_D \cdot S \Rightarrow \pi_D \cdot T$. Then we define a displayed bicategory $\mathsf{DFcell}(l, r)$ over $\int D$ such that

- the objects over x are 2-cells $\gamma_x : l(x) \Rightarrow r(x)$;
- the 1-cells over $f : x \to y$ from γ_x to γ_y are equalities

$$(\gamma_x \rhd T(\pi_D(f))) \bullet r(f) = l(f) \bullet (S(\pi_D(f)) \lhd \gamma_y);$$

- the 2-cells over $\theta : f \Rightarrow g$ are inhabitants of the unit type.

Example 2.10. Let B be a bicategory and let P be a family of propositions on the objects of B. Then we define a displayed bicategory $\mathsf{FSub}(P)$ over B whose objects over x are proofs of $P(x)$ and whose displayed 1-cells and 2-cells are inhabitants of the unit type. The total bicategory $\int \mathsf{FSub}(P)$ is the **full subbicategory** of B whose objects satisfy P.

3 Signatures and their Algebras

Before we can discuss how to construct 1-truncated higher inductive types, we need to define signatures for those. Our notion of signature is similar to the one by Dybjer and Moenclaey [28]. However, instead of defining them externally, we define a type of signatures within type theory just like what was done for inductive-recursive and inductive-inductive definitions [29, 32]. In addition, we show that each signature Σ gives rise to a bicategory of algebras for Σ.

In this section, we study HITs of the following shape

$$\begin{aligned}
&\mathsf{Inductive}\ H :=\\
&\mid\ c : P(H) \to H\\
&\mid\ p : \textstyle\prod(j : I)(x : Q(H)), l(x) = r(x)\\
&\mid\ s : \textstyle\prod(j : J)(x : R(H))(r : a_1(x) = a_2(x)),\\
&\qquad q_1(x, r) = q_2(x, r)\\
&\mid\ t : \textstyle\prod(x, y : H)(q_1, q_2 : x = y)(r, s : q_1 = q_2), r = s
\end{aligned}$$

To see what the challenges are when defining such HITs, let us take a closer look at the torus.

$$\begin{aligned}
&\mathsf{Inductive}\ \mathcal{T}^2 :=\\
&\mid\ \mathbf{base} : \mathcal{T}^2\\
&\mid\ \mathbf{loop_l}, \mathbf{loop_r} : \mathbf{base} = \mathbf{base}\\
&\mid\ \mathbf{surf} : \mathbf{loop_l} \bullet \mathbf{loop_r} = \mathbf{loop_r} \bullet \mathbf{loop_l}
\end{aligned}$$

There is a point constructor **base**, two paths constructors $\mathbf{loop_l}, \mathbf{loop_r} : \mathbf{base} = \mathbf{base}$, and a homotopy constructor $\mathbf{surf} : \mathbf{loop_l} \bullet \mathbf{loop_r} = \mathbf{loop_r} \bullet \mathbf{loop_l}$. Note that $\mathbf{loop_l}$ and $\mathbf{loop_r}$ refer to **base** and that **surf** refers to all other constructors. Hence, the signatures we define, must be flexible enough to allow such dependencies.

A similar challenge comes up when defining the bicategory of algebras for a signature. For the torus, an algebra would consist of a type X, a point b, paths $p, q : b = b$, and a 2-path $s : p \bullet q = q \bullet p$. Again there are dependencies: p and q depend on b while s depends on both p and q. To overcome this challenge, we use displayed bicategories to construct the bicategory of algebras in a stratified way.

3.1 Signatures

Now let us define signatures, and to do so, we must specify data which describes the constructors for points, paths, and homotopies. To specify the point constructors, we use *polynomial codes*. Given a type X and a polynomial code P, we get another type $P(X)$. Such a code P describes an operation of the form $P(X) \to X$.

Definition 3.1. The type of **codes for polynomials** is inductively generated by the following constructors

$$\mathsf{C}(A) : \mathsf{P}, \quad \mathsf{Id} : \mathsf{P}, \quad P_1 + P_2 : \mathsf{P}, \quad P_1 \times P_2 : \mathsf{P}$$

where A is a 1-type and P_1 and P_2 are elements of P.

The constructor $\mathsf{C}(A)$ represents the constant polynomial, Id represents the identity, and $P_1 + P_2$ and $P_1 \times P_2$ represent the sum and product respectively. Note that we restrict ourselves to finitary polynomials since we do not have a constructor which represents the function space.

The second part of the signature describes the path constructors, which represent universally quantified equations. To describe them, we must give two *path endpoints*. These endpoints can refer to the point constructor, which we represent by a polynomial A. In addition, they have a source (the type of the quantified variable) and a target (the type of the term). The source and the target are represented by polynomials S and T respectively.

Definition 3.2. Let A, S, and T be codes for polynomials. The type $\mathsf{E}_A(S, T)$ of **path endpoints** with arguments A, source S, and target T is inductively generated by the constructors given in Figure 2.

Note that the parameter A is only used in the path endpoint **constr**, which represents the point constructor. If we have

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

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

$$\frac{P : \mathsf{P} \qquad X : \text{1-Type} \qquad x : X}{\mathbf{c}(x) : \mathsf{E}_A(P, \mathsf{C}(X))} \qquad \frac{P, Q, R : \mathsf{P} \qquad e_1 : \mathsf{E}_A(P, Q) \qquad e_2 : \mathsf{E}_A(P, R)}{(e_1, e_2) : \mathsf{E}_A(P, Q \times R)}$$

Figure 2. Rules for the path endpoints.

$$\frac{T : \mathsf{P} \qquad e : \mathsf{E}_A(R, T)}{\mathbf{idpath}(e) : \mathsf{H}_{l,r,a_1,a_2}(e, e)} \qquad \frac{T : \mathsf{P} \qquad e_1, e_2 : \mathsf{E}_A(R, T) \qquad h : \mathsf{H}_{l,r,a_1,a_2}(e_1, e_2)}{h^{-1} : \mathsf{H}_{l,r,a_1,a_2}(e_2, e_1)}$$

$$\frac{T : \mathsf{P} \qquad e_1, e_2, e_3 : \mathsf{E}_A(R, T) \qquad h_1 : \mathsf{H}_{l,r,a_1,a_2}(e_1, e_2) \qquad h_2 : \mathsf{H}_{l,r,a_1,a_2}(e_2, e_3)}{h_1 \mathbin{@} h_2 : \mathsf{H}_{l,r,a_1,a_2}(e_1, e_3)}$$

$$\frac{T_1, T_2, T_3 : \mathsf{P} \qquad e_1 : \mathsf{E}_A(R, T_1) \qquad e_2 : \mathsf{E}_A(T_1, T_2) \qquad e_3 : \mathsf{E}_A(T_2, T_3)}{\boldsymbol{\alpha}(e_1, e_2, e_3) : \mathsf{H}_{l,r,a_1,a_2}(e_1 \cdot (e_2 \cdot e_3), (e_1 \cdot e_2) \cdot e_3)}$$

$$\frac{T : \mathsf{P} \qquad e : \mathsf{E}_A(R, T)}{\boldsymbol{\lambda}(e) : \mathsf{H}_{l,r,a_1,a_2}(\mathbf{id}_R \cdot e, e)} \qquad \frac{T : \mathsf{P} \qquad e : \mathsf{E}_A(R, T)}{\boldsymbol{\rho}(e) : \mathsf{H}_{l,r,a_1,a_2}(e \cdot \mathbf{id}_T, e)}$$

$$\frac{T_1, T_2 : \mathsf{P} \qquad e_1, e_2 : \mathsf{E}_A(R, T_1) \qquad e_3, e_4 : \mathsf{E}_A(R, T_2) \qquad h : \mathsf{H}_{l,r,a_1,a_2}((e_1, e_3), (e_2, e_4))}{\mathbf{pr}_1(h) : \mathsf{H}_{l,r,a_1,a_2}(e_1, e_2), \quad \mathbf{pr}_2(h) : \mathsf{H}_{l,r,a_1,a_2}(e_3, e_4)}$$

$$\frac{T_1, T_2 : \mathsf{P} \qquad e_1, e_2 : \mathsf{E}_A(R, T_1) \qquad e_3, e_4 : \mathsf{E}_A(R, T_2) \qquad h_1 : \mathsf{H}_{l,r,a_1,a_2}(e_1, e_2) \qquad h_2 : \mathsf{H}_{l,r,a_1,a_2}(e_3, e_4)}{(h_1, h_2) : \mathsf{H}_{l,r,a_1,a_2}((e_1, e_3), (e_2, e_4))}$$

$$\frac{T_1, T_2 : \mathsf{P} \qquad e_1, e_2 : \mathsf{E}_A(R, T_1) \qquad h : \mathsf{H}_{l,r,a_1,a_2}(e_1, e_2)}{\mathbf{inl}(h) : \mathsf{H}_{l,r,a_1,a_2}(e_1 \cdot \mathbf{inl}, e_2 \cdot \mathbf{inl})} \qquad \frac{j : J \qquad e : \mathsf{E}_A(R, Q_j)}{\mathbf{path}_j(e) : \mathsf{H}_{l,r,a_1,a_2}(e \cdot l(j), e \cdot r(j))}$$

$$\frac{T_1, T_2 : \mathsf{P} \qquad e_1, e_2 : \mathsf{E}_A(R, T_2) \qquad h : \mathsf{H}_{l,r,a_1,a_2}(e_1, e_2)}{\mathbf{inr}(h) : \mathsf{H}_{l,r,a_1,a_2}(e_1 \cdot \mathbf{inr}, e_2 \cdot \mathbf{inr})} \qquad \mathbf{p}_{\mathrm{arg}} : \mathsf{H}_{l,r,a_1,a_2}(a_1, a_2)$$

$$\frac{e_l, e_r : \mathsf{E}_A(Q, A) \qquad h : \mathsf{H}_{l,r,a_1,a_2}(e_l, e_r)}{\mathbf{ap}\ h : \mathsf{H}_{l,r,a_1,a_2}(e_l \cdot \mathbf{constr}, e_r \cdot \mathbf{constr})}$$

Figure 3. Rules for the homotopy endpoints.

a type X with a function $c : A(X) \to X$, then each endpoint e gives for every $x : S(X)$ a point $[\![e]\!](x) : T(X)$. Note that $[\![e]\!](x)$ depends on c while we do not write c in the notation. Hence, two endpoints e_l, e_r represent the equation

$$\prod(x : S(X)), [\![e_l]\!](x) = [\![e_r]\!](x).$$

Note that a HIT could have arbitrarily many path constructors and we index them by the type J.

The last part of the signature describes the homotopy constructors and these depend on both the point and path constructors. A homotopy constructor represents an equation of paths, which is universally quantified over both points and paths of the HIT being defined. The point argument is represented by a polynomial R, and the path argument is represented by a polynomial T and endpoints $a_1, a_2 : \mathsf{E}_A(R, T)$. Lastly, the type of the paths in the equation is described by two endpoints $s_l, s_r : \mathsf{E}_A(R, W)$ with $W : \mathsf{P}$.

Definition 3.3. Suppose that we have

- A polynomial A;
- A type J together with for each $j : J$ a polynomial Q_j and endpoints $l_j, r_j : \mathsf{E}_A(Q_j, \mathsf{Id})$;
- A polynomial R;
- A polynomial T with endpoints $a_1, a_2 : \mathsf{E}_A(R, T)$;
- A polynomial W with endpoints $s_1, s_2 : \mathsf{E}_A(R, W)$.

Then we define the type $\mathsf{H}_{l,r,a_1,a_2}(s_1, s_2)$ of **homotopy endpoint** inductively by the constructors in Figure 3.

There are three homotopy endpoints of particular importance. The first one is **path**, which represents the path constructor and it makes use of l_j and r_j. The second one, $\mathbf{p}_{\mathrm{arg}}$, represents the path argument and it uses a_1 and a_2. The last

one is **ap** and it represents the action of the point constructor on a homotopy endpoint.

The way we represent path arguments allows us to represent equations with any finite number of path arguments by only two path endpoints. For example, two path arguments $p : x_1 = y_1$ and $q : x_2 = y_2$ is represented by one path argument of type $(x_1, x_2) = (y_1, y_2)$.

Given a type X with a function $c : A(X) \to X$ and for each $x : Q_j(X)$ a path $l_j(x) = r_j(x)$, a homotopy endpoint $p : \mathsf{H}_{l,r,a_1,a_2}(s_1, s_2)$ gives rise for each point $x : R(X)$ and path $w : a_1(x) = a_2(x)$ to another path $p(x, w) : s_1(x) = s_2(x)$. Hence, two homotopy endpoints $p, q : \mathsf{H}_{l,r,a_1,a_2}(s_1, s_2)$ represent the equation

$$\prod(x : R(X))(w : a_1(x) = a_2(x)), p(x, w) = q(x, w)$$

Now let us put this all together and define what signatures for higher inductive types are.

Definition 3.4. A **HIT-signature** Σ consists of

- A polynomial A^Σ;
- A type $\mathsf{J}_\mathsf{P}^\Sigma$ together with for each $j : \mathsf{J}_\mathsf{P}^\Sigma$ a polynomial S_j^Σ and endpoints $\mathsf{l}_j^\Sigma, \mathsf{r}_j^\Sigma : \mathsf{E}_{\mathsf{A}^\Sigma}(\mathsf{S}_j^\Sigma, \mathsf{Id})$;
- A type $\mathsf{J}_\mathsf{H}^\Sigma$ together with for each $j : \mathsf{J}_\mathsf{H}^\Sigma$ polynomials R_j^Σ and T_j^Σ, endpoints $\mathsf{a}_j^\Sigma, \mathsf{b}_j^\Sigma : \mathsf{E}_{\mathsf{A}^\Sigma}(\mathsf{R}_j^\Sigma, \mathsf{T}_j^\Sigma)$ and $\mathsf{s}_j^\Sigma, \mathsf{t}_j^\Sigma : \mathsf{E}_{\mathsf{A}^\Sigma}(\mathsf{R}_j^\Sigma, \mathsf{Id})$, and homotopy endpoints $\mathsf{p}_j^\Sigma, \mathsf{q}_j^\Sigma : \mathsf{H}_{\mathsf{l}_j^\Sigma, \mathsf{r}_j^\Sigma, \mathsf{a}_j^\Sigma, \mathsf{b}_j^\Sigma}(\mathsf{s}_j^\Sigma, \mathsf{t}_j^\Sigma)$.

If Σ is clear from the context, we do not write the superscript. In the remainder, we show how to interpret the following HIT given a signature Σ:

```
Inductive H :=
| c : A(H) → H
| p : ∏(j : J_P)(x : S_j(H)), [[l_j]](x) = [[r_j]](x)
| s : ∏(j : J_H)(x : R_j(H))(r : a_j(x) = b_j(x)),
      p_j(x, r) = q_j(x, r)
| t : ∏(x, y : H)(p, q : x = y)(r, s : p = q), r = s
```

Next we consider three examples of HITs we can express with these signatures.

Example 3.5. The torus is described by the signature $\mathcal{T}^2$.

- Take $\mathsf{A}^{\mathcal{T}^2} = \mathbf{C}(1)$;
- Take $\mathsf{J}_\mathsf{P}^{\mathcal{T}^2} = 2$ and for both inhabitants we take $\mathsf{S}^{\mathcal{T}^2} = \mathbf{C}(1)$ and $\mathsf{l}^{\mathcal{T}^2} = \mathsf{r}^{\mathcal{T}^2} = \mathbf{constr}$;
- Take $\mathsf{J}_\mathsf{H}^{\mathcal{T}^2} = 1$. Since there no arguments for this path constructor, we take $\mathsf{R}^{\mathcal{T}^2} = \mathbf{C}(1)$ and $\mathsf{a}^{\mathcal{T}^2} = \mathsf{b}^{\mathcal{T}^2} = \mathbf{c}(\mathbf{tt})$. Now for the left-hand side and right-hand side of this equation, we take $\mathbf{path}_{\mathbf{true}}(\mathbf{id}) \mathbin{@} \mathbf{path}_{\mathbf{false}}(\mathbf{id})$ and $\mathbf{path}_{\mathbf{false}}(\mathbf{id}) \mathbin{@} \mathbf{path}_{\mathbf{true}}(\mathbf{id})$ respectively.

Example 3.6. We represent the integers modulo 2 as the following HIT:

```
Inductive ℤ_2 :=
| Z : ℤ_2
| S : ℤ_2 → ℤ_2
| m : ∏(x : ℤ_2), S(S(x)) = x
| c : ∏(x : ℤ_2), m(S(x)) = ap S (m(x))
```

Note that all constructors except **Z** are recursive. We define a signature $\mathbb{Z}_2$.

- Take $\mathsf{A}^{\mathbb{Z}_2} = \mathbf{C}(1) + \mathsf{Id}$;
- Take $\mathsf{J}_\mathsf{P}^{\mathbb{Z}_2} = \mathbf{C}(1)$ and for its unique inhabitant we take $\mathsf{S}^{\mathbb{Z}_2} = \mathsf{Id}$ and
$$\mathsf{l}^{\mathbb{Z}_2} = (\mathbf{inr} \cdot \mathbf{constr}) \cdot (\mathbf{inr} \cdot \mathbf{constr}), \quad \mathsf{r}^{\mathbb{Z}_2} = \mathbf{id};$$
- Take $\mathsf{J}_\mathsf{H}^{\mathbb{Z}_2} = \mathbf{C}(1)$. Furthermore, we take $\mathsf{R}^{\mathbb{Z}_2} = \mathsf{Id}$ and $\mathsf{a}^{\mathbb{Z}_2} = \mathsf{b}^{\mathbb{Z}_2} = \mathbf{c}(\mathbf{tt})$. The endpoints s and t encode $\mathbf{S}(\mathbf{S}(\mathbf{S}(x)))$ and $\mathbf{S}(x)$ respectively, and for the left-hand side and right-hand side of this equation, we take
$$\mathbf{ap}\,(\boldsymbol{\lambda}^{-1} \mathbin{@} \boldsymbol{\alpha} \mathbin{@} \mathbf{inr}(\mathbf{path}_1(\mathbf{id})) \mathbin{@} \boldsymbol{\alpha}^{-1} \mathbin{@} \boldsymbol{\lambda} \mathbin{@} \boldsymbol{\lambda})$$
$$\boldsymbol{\alpha}^{-1} \mathbin{@} \boldsymbol{\alpha}^{-1} \mathbin{@} \mathbf{path}_{\mathbf{tt}}(\mathbf{inr} \cdot \mathbf{constr}) \mathbin{@} \boldsymbol{\rho}.$$
respectively. Note that we use $\boldsymbol{\alpha}$, $\boldsymbol{\lambda}$, and $\boldsymbol{\rho}$ to make the equations type check. If we would interpret the left-hand side and right-hand side of the homotopy constructor in 1-types, then all occurrences of $\boldsymbol{\alpha}$, $\boldsymbol{\lambda}$, and $\boldsymbol{\rho}$ become the identity path. We thus get the right homotopy constructor.

Example 3.7. Given a 1-type A, the set truncation of A is defined by the following HIT:

```
Inductive ||A|| :=
| inc : A → ||A||
| trunc : ∏(x, y : ||A||)(p, q : x = y), p = q
```

Note that this higher inductive type has a parameter A, so the signature we define, depends on a 1-type A as well. To encode the path arguments of **trunc**, we use that two paths $p, q : x = y$ is the same as a path $r : (x, x) = (y, y)$. Define a signature $||A||$ such that

- $\mathsf{A}^{||A||} = \mathbf{C}(A)$;
- $\mathsf{J}_\mathsf{P}^{||A||}$ is the empty type;
- $\mathsf{J}_\mathsf{H}^{||A||} = 1$. In addition, there are two point arguments $\mathsf{R}^{||A||} = \mathsf{Id} \times \mathsf{Id}$ and a path argument with left-hand side $(\mathbf{pr}_1, \mathbf{pr}_1)$ and right-hand side $(\mathbf{pr}_2, \mathbf{pr}_2)$. For the left-hand side and right-hand side of the homotopy, we take $\mathbf{pr}_1(\mathbf{p}_{\mathrm{arg}})$ and $\mathbf{pr}_2(\mathbf{p}_{\mathrm{arg}})$ respectively.

3.2 Algebras in 1-types and groupoids

With the signatures in place, our next goal is to study the introduction rules of HITs and for that, we define bicategories of algebras for a signature. Since we ultimately want to construct HITs via the groupoid quotient, we look at both algebras in 1-types and groupoids.

In both cases, we use a stratified approach with displayed bicategories. Let us illustrate this by briefly describing the construction for 1-types. On 1-Type, we define a displayed bicategory and we denote its total bicategory by PreAlg(Σ).

The objects of $\mathsf{PreAlg}(\Sigma)$ consist of a 1-type X together with an operation $P(X) \to X$. Concretely, the objects satisfy the point introduction rules specified by Σ. On top of $\mathsf{PreAlg}(\Sigma)$, we define another displayed bicategory whose total bicategory is denoted by $\mathsf{PathAlg}(\Sigma)$. Objects of $\mathsf{PathAlg}(\Sigma)$ satisfy the introduction rules for both the points and the paths. Lastly, we take a full subbicategory of $\mathsf{PathAlg}(\Sigma)$ obtaining another bicategory $\mathsf{Alg}(\Sigma)$ whose objects satisfy the introduction rules for the points, paths and homotopies.

To define $\mathsf{PreAlg}(\Sigma)$, we use Example 2.7.

Problem 3.8. *Given $P : \mathsf{P}$, to construct pseudofunctors*

$$[\![P]\!] : \mathsf{Pseudo}(\text{1-Type}, \text{1-Type}), \langle P \rangle : \mathsf{Pseudo}(\mathsf{Grpd}, \mathsf{Grpd}).$$

Construction 3.9 (for Problem 3.8)**.** We only discuss the case for 1-types since the case for groupoids is similar. Given a polynomial P and a type X, we get a type $P(X)$ by induction. The verification that this gives rise to a pseudofunctor can be found in the formalization. □

Definition 3.10. Let Σ be a signature. Then we define the bicategories $\mathsf{PreAlg}(\Sigma)$ and $\mathsf{PreAlg}_{\mathsf{Grpd}}(\Sigma)$ to be the total bicategories of $\mathsf{DFalg}([\![\mathsf{A}^\Sigma]\!])$ and $\mathsf{DFalg}(\langle \mathsf{A}^\Sigma \rangle)$ respectively. Objects of these bicategories are called **prealgebras** for Σ.

Note that prealgebras only have structure witnessing the introduction rule for the points. Next we look at the introduction rule for the paths. In this case, the desired structure is added via Example 2.9 and to apply this construction, we interpret path endpoints as pseudotransformations.

Problem 3.11. *Given $e : \mathsf{E}_A(P, Q)$, to construct pseudotransformations*

$$[\![e]\!] : [\![P]\!] \cdot \pi_{\mathsf{DFalg}([\![A]\!])} \Rightarrow [\![Q]\!] \cdot \pi_{\mathsf{DFalg}([\![A]\!])},$$
$$\langle e \rangle : \langle P \rangle \cdot \pi_{\mathsf{DFalg}(\langle A \rangle)} \Rightarrow \langle Q \rangle \cdot \pi_{\mathsf{DFalg}(\langle A \rangle)}.$$

Construction 3.12 (for Problem 3.11)**.** We only discuss $[\![e]\!]$ since $\langle e \rangle$ is defined similarly. Given a 1-type X and $c : A(X) \to X$, we define the function $[\![e]\!] : P(X) \to Q(X)$ by induction. The verification that this gives rise to a pseudotransformation can be found in the formalization. □

Definition 3.13. Let Σ be a signature. We use Examples 2.8 and 2.9 to define displayed bicategories over $\mathsf{PreAlg}(\Sigma)$ and $\mathsf{PreAlg}_{\mathsf{Grpd}}(\Sigma)$.

$$\mathsf{DPathAlg}(\Sigma) = \prod (i : \mathsf{J}_\mathsf{P}^\Sigma), \mathsf{DFcell}([\![\mathsf{l}^\Sigma(i)]\!], [\![\mathsf{r}^\Sigma(i)]\!])$$
$$\mathsf{DPathAlg}_{\mathsf{Grpd}}(\Sigma) = \prod (i : \mathsf{J}_\mathsf{P}^\Sigma), \mathsf{DFcell}(\langle \mathsf{l}^\Sigma(i) \rangle, \langle \mathsf{r}^\Sigma(i) \rangle)$$

We define $\mathsf{PathAlg}(\Sigma)$ and $\mathsf{PathAlg}_{\mathsf{Grpd}}(\Sigma)$ to be the total bicategories of $\mathsf{DPathAlg}(\Sigma)$ and $\mathsf{DPathAlg}_{\mathsf{Grpd}}(\Sigma)$ respectively. Objects of $\mathsf{PathAlg}(\Sigma)$ and $\mathsf{PathAlg}_{\mathsf{Grpd}}(\Sigma)$ are called **path algebras** for Σ.

Problem 3.14. *Suppose that we have a homotopy endpoint $h : \mathsf{H}_{l,r,a_1,a_2}(s_1, s_2)$. Given a 1-type X with $c : A(X) \to X$ and $p : \prod (j : J)(x : Q_j(X)), l_j(x) = r_j(x)$, to construct for each $x : Q(X)$ and $w : [\![a_1]\!](x) = [\![a_2]\!](x)$ an equality $[\![h]\!](x, w) : [\![s_1]\!](x) = [\![s_2]\!](x)$.*

In addition, given a groupoid G together with a functor $c : \langle A \rangle(G) \to G$ and for each $j : J$ a natural transformation $\langle l_j \rangle(G) \Rightarrow \langle r_j \rangle(G)$, to construct for each object $x : \langle Q \rangle(G)$ and morphism $w : \langle a_1 \rangle(G)(x) \to \langle a_2 \rangle(G)(x)$ a morphism $\langle h \rangle(x, w) : \langle s_1 \rangle(G)(x) \to \langle s_2 \rangle(G)(x)$.

Construction 3.15 (for Problem 3.14)**.** By induction. □

Definition 3.16. Let Σ be a HIT signature. We define $\mathsf{Alg}(\Sigma)$ to be the full subbicategory of $\mathsf{PathAlg}(\Sigma)$ in which every object X satisfies

$$\prod (j : \mathsf{J}_\mathsf{H}^\Sigma)(x : \mathsf{R}_j^\Sigma(X))(w : [\![\mathsf{a}_j^\Sigma]\!](x) = [\![\mathsf{b}_j^\Sigma]\!](x)), \\ [\![\mathsf{p}_j^\Sigma]\!](x, w) = [\![\mathsf{q}_j^\Sigma]\!](x, w)$$

In addition, we define $\mathsf{Alg}_{\mathsf{Grpd}}(\Sigma)$ to be the full subbicategory of $\mathsf{PathAlg}_{\mathsf{Grpd}}(\Sigma)$ in which every object X satisfies

$$\prod (j : \mathsf{J}_\mathsf{H}^\Sigma)(x : \langle \mathsf{R}_j^\Sigma \rangle X)(w : \langle \mathsf{a}_j^\Sigma \rangle(x) = \langle \mathsf{b}_j^\Sigma \rangle(x)), \\ \langle \mathsf{p}_j^\Sigma \rangle(x, w) = \langle \mathsf{q}_j^\Sigma \rangle(x, w)$$

Objects of $\mathsf{Alg}(\Sigma)$ and $\mathsf{Alg}_{\mathsf{Grpd}}(\Sigma)$ are called **algebras** for Σ.

The bicategory $\mathsf{Alg}(\Sigma)$ is constructed by repeatedly using Definition 2.6. By unpacking the definition, we see that an algebra $X : \mathsf{Alg}(\Sigma)$ consists of

- A 1-type X;
- A function $\mathsf{c}^X : \mathsf{A}(X) \to X$;
- For each $j : \mathsf{J}_\mathsf{P}$ and point $x : \mathsf{S}_j(X)$ a path $\mathsf{p}_j^X(x) : [\![\mathsf{l}_j]\!](x) = [\![\mathsf{r}_j]\!](x)$;
- For each $j : \mathsf{J}_\mathsf{H}$, $x : \mathsf{R}(X)$ and $w : [\![a_1]\!](x) = [\![a_2]\!](x)$, a homotopy $\mathsf{h}_j^X : [\![\mathsf{p}]\!](x, w) = [\![\mathsf{q}]\!](x, w)$

4 Induction and Biinitiality

The algebra structure only represents the introduction rule and the next step is to define the elimination and computation rules for higher inductive types. Before we can formulate these principles, we need to define dependent actions of polynomials, path endpoints, and homotopy endpoints. All of these constructions are done by induction and details can be found in the literature [28, 35, 61].

Problem 4.1. *Given a type X, a type family Y on X, and a polynomial P, to construct a type family $\overline{P}(Y)$ on $P(Y)$.*

Problem 4.2. *Given a type X, a type family Y on X, a polynomial P, and a map $f : \prod (x : X), Y(x)$, to construct a map $\overline{P}(f) : \prod (x : P(X)), \overline{P}(Y)(x)$,*

Problem 4.3. *Given a type X, a type family Y on X, an endpoint $e : \mathsf{E}_A(P, Q)$, and a map $c : A(X) \to X$, to construct for each $x : P(X)$ and $y : \overline{P}(Y)(x)$ an inhabitant $[\![\overline{e}]\!](y) : \overline{Q}(Y)([\![e]\!](x))$.*

Problem 4.4. *Suppose, that we have polynomials A, P, Q, a type X with a map $c_X : A(X) \to X$, and a type family Y on X with a map $c_Y : \prod(x : X), \overline{A}(Y)(x) \to Y(c_X(x))$ and a map $f : \prod(x : X), Y(x)$. Given an endpoint $e : \mathsf{E}_A(P, Q)$, to construct an equality*

$$[\![\overline{e}]\!](f) : \overline{Q}(f)([\![e]\!](x)) = [\![\overline{e}]\!](\overline{P}(f)(x)).$$

Problem 4.5. *Let Σ be a signature. Let X be a type with a function $c_X : \mathsf{A}(X) \to X$ and with for each $j : \mathsf{J_P}$ and $x : \mathsf{S}(X)$ a path $p_X(j, x) : [\![\mathsf{l}]\!](x) = [\![\mathsf{r}]\!](x)$. In addition, suppose that Y is a type family on X, that we have a function $c_Y : \prod(x : \mathsf{A}(X)), \overline{\mathsf{A}}(Y)(x) \to Y(c_X(x))$, and that for all $j : \mathsf{J_P}$ and points $x : \mathsf{S}(X)$ and $\overline{x} : \overline{\mathsf{S}}(Y)(x)$, we have a path $p_Y : [\![\overline{\mathsf{l}}]\!](\overline{x}) =_{p_X(j,x)} [\![\overline{\mathsf{r}}]\!](\overline{x})$. Furthermore, let $x : \mathsf{R}$ and $\overline{x} : \overline{\mathsf{R}}(Y)(x)$ be points and let $w : [\![\mathsf{a}]\!](x) = [\![\mathsf{b}]\!](x)$ and $\overline{w} : [\![\overline{\mathsf{a}}]\!](\overline{x}) =_p [\![\overline{\mathsf{b}}]\!](\overline{x})$ be paths. Then for each homotopy endpoint $h : \mathsf{H}_{l,r,a_1,a_2}(s_1, s_2)$, to construct a path*

$$\overline{h}(\overline{x}, \overline{w}) : [\![\overline{\mathsf{s}}]\!](\overline{x}) =_{h(x,w)} [\![\overline{\mathsf{t}}]\!](\overline{x}).$$

With these notions in place, we define *displayed algebras*. These represent the input of the elimination rule.

Definition 4.6. Given a signature Σ and an algebra X for Σ, a **displayed algebra** Y over X consists of

- A family Y of 1-types over X;
- For each $x : \mathsf{A}(X)$ a map $\overline{\mathsf{c}}^Y : \overline{\mathsf{A}}(Y)(x) \to Y(\mathsf{c}^X(x))$;
- For each $j : \mathsf{J_P}$, $x : \mathsf{S}_j(X)$, and $\overline{x} : \overline{\mathsf{S}_j}(Y)(x)$, a path $\overline{\mathsf{p}}_j^Y : [\![\overline{l_j}]\!](\overline{x}) =_{\mathsf{p}_j^X x} [\![\overline{r_j}]\!](\overline{x})$;
- For each $j : \mathsf{J_H}$, points $x : \mathsf{R}(X)$ and $\overline{x} : \overline{\mathsf{R}}(Y)(x)$, and paths $w : \mathsf{a}(x) = \mathsf{b}(x)$ and $\overline{w} : [\![\overline{\mathsf{a}}]\!](\overline{x}) =_w [\![\overline{\mathsf{b}}]\!](\overline{x})$, a globe $\overline{\mathsf{h}}_j^Y : \overline{\mathsf{p}}(\overline{x}) =_{\mathsf{h}_j^X(x,w)} \overline{\mathsf{q}}(\overline{x})$ over $\mathsf{h}_j^X(x, w)$.

Remark 4.7. The type family of a displayed algebra is required to be 1-truncated. This means that the HITs we construct, can only be eliminated into 1-types, and as a consequence, these HITs only have the right elimination principle with respect to 1-types.

The output of the elimination rule and the computation rules are given by a *section* to be defined in Definition 4.9 below. One might expect that, just like for the groupoid quotient, the computation rules for the paths are given as globes over some 2-path in the base (Definition 2.2). However, this is not the case.

This is because there is a slight discrepancy between the rules for the groupoid quotient and the HITs we discuss, namely for the former the computation rules for the points are definitional equalities while for the latter, these rules only hold propositionally. This affects how we need to formulate the computation rules for the paths.

Let us illustrate this via the torus (Example 3.5). The input for the elimination rule consists, among others, of a type family Y, a point $b : Y(\mathbf{base})$, and a path $p_l : b =_{\mathbf{loop_l}} b$. The elimination rule gives a map $f : \prod(x : \mathcal{T}^2), Y(x)$. By the point computation rule, we have a propositional equality between $f(\mathbf{base})$ and b. Now the computation rule for $\mathbf{loop_l}$ ought to equate apd f $\mathbf{loop_l}$ and p_l. However, such an equation does not type check since apd f $\mathbf{loop_l}$ has type $f(b) =_{\mathbf{loop_l}} f(b)$ while p_l has type $b =_{\mathbf{loop_l}} b$. In conclusion, we cannot formulate the computation rules the same way as we did for the grorupoid quotient.

Our solution to this problem is to define a type of *squares* over a given 2-path similarly to Definition 2.2.

Definition 4.8. Let X be a type and let Y be a type family on X. Suppose that we are given points $x_1, x_2 : X$ and $\overline{x_1}, \overline{x_1}' : Y(x_1)$ and $\overline{x_2}, \overline{x_2}' : Y(x_2)$, paths $p, q : x_1 = x_2$ together with paths $\overline{p} : \overline{x_1} =_p \overline{x_2}$ and $\overline{q} : \overline{x_1}' =_q \overline{x_2}'$ over p and q respectively. If we also have two paths $h_1 : \overline{x_1} = \overline{x_1}'$ and $h_2 : \overline{x_2} = \overline{x_2}'$ and a 2-path $g : p = q$, then we define the type of **squares** over g from $\overline{p}$ to $\overline{q}$ with sides h_1 and h_2 by path induction.

Definition 4.9. Let X be an algebra for a given signature Σ and let Y be a displayed algebra over X. Then a **section** of Y consists of

- A map $f : \prod(x : X), Y(x)$;
- For all $x : \mathsf{A}(X)$, an equality $f(\mathsf{c}^X(x)) = \overline{\mathsf{c}}^Y(\overline{\mathsf{A}}(f)(x))$;
- For all $j : \mathsf{J_P}$ and $x : \mathsf{S}(X)$, a square from apd f $(\mathsf{p}_j^X(x))$ to $\overline{\mathsf{p}}_j^Y(\overline{\mathsf{S}}(f)(x))$ with sides $[\![\overline{\mathsf{l}}]\!](f)(x)$ and $[\![\overline{\mathsf{r}}]\!](f)(x)$.

Definition 4.10. Let Σ be a signature and let X be an algebra for Σ. Then we say that X is a **1-truncated higher inductive type** for Σ if each displayed algebra Y over X has a section.

Often we just say that X is a HIT for Σ instead of saying that X is 1-truncated HIT. With this in place, we can check whether our rules for higher inductive types agree for the usual examples with the rules given in the literature [59]. We illustrate this with the torus (Example 3.5) and the set truncation (Example 3.7). In the next example, we write $p \bullet q$ for the concatenation of dependent paths.

Example 4.11 (Example 3.5 cont'd)**.** Recall the signature $\mathcal{T}^2$ for the torus. Let X be a HIT for $\mathcal{T}^2$. Since X is an algebra, we have a point $\mathbf{base} : X$, two paths $\mathbf{loop_l}, \mathbf{loop_r} : \mathbf{base} = \mathbf{base}$, and a 2-path $\mathbf{surf} : \mathbf{loop_l} \bullet \mathbf{loop_r} = \mathbf{loop_r} \bullet \mathbf{loop_l}$. This corresponds precisely to the usual introduction rules of the torus.

A family Y of 1-types on X together with a point $b : Y(\mathbf{base})$, two paths $l : b =_{\mathbf{loop_l}} b$ and $r : b =_{\mathbf{loop_r}} b$ and a globe $h : l \bullet r =_{\mathbf{surf}} r \bullet l$ over $\mathbf{surf}$ gives rise to a displayed algebra over X. This corresponds to the usual input of the elimination rule of the torus. If we have a section s of Y, then in particular, we get a map $f_s : \prod(x : X), Y(x)$. We also get a path $p_s : f(\mathbf{base}) = b$, a square from apd f $\mathbf{loop_l}$ to l and one from apd f $\mathbf{loop_r}$ to r. Both squares have sides p_s and p_s. These are the computation rules for the points and paths

of the torus. Note that since we are looking at 1-truncated HITs, this only gives the 1-truncation of the torus.

Example 4.12 (Example 3.7 cont'd)**.** Let A be a 1-type and recall the signature $||A||$. Now let X be a HIT on $||A||$. Note that an algebra for $||A||$ consists of a type Z together with a map $A \to Z$ and a proof that Z is a set. This means in particular, that we have a map $\mathbf{inc} : A \to X$ and a proof **trunc** that X is a set.

A family Y of sets on X together with a map $i : \prod(a : A), Y(\mathbf{inc}(A))$ give rise to a displayed algebra over X. A section s of that displayed algebra consists of a map $f_s : \prod(x : X), Y(x)$ such that $f_s(\mathbf{inc}(a)) = i(a)$ for all $a : A$. This corresponds to the usual elimination and computation rules for the set truncation.

To verify that an algebra satisfies the elimination rule, we use *initial algebra semantics* [35]. However, this technique is usually applied in a categorical setting and it uses initial objects in categories. Since we work in a bicategorical setting, we need to use the corresponding notion in bicategory theory: *biinitiality*.

Definition 4.13. Let B be a bicategory and let x be an object in B. Then we say x is **biinitial** if

- For each object y there is a 1-cell $x \to y$;
- Given 1-cells $f, g : x \to y$, there is a 2-cell $f \Rightarrow g$;
- Given 2-cells $\theta, \theta' : f \Rightarrow g$, there is an equality $\theta = \theta'$.

Briefly, an object x is biinitial if for each y there is 1-cell from x to y, which is unique up to a unique 2-cell. Now we can formulate initial algebra semantics for our signatures.

Proposition 4.14. *Let Σ be a signature and let X be an algebra for Σ. Then*

- *If X is a 1-truncated HIT for Σ, then X is biinitial in* $\mathrm{Alg}(\Sigma)$.
- *If X is biinitial, then X is a 1-truncated HIT for Σ.*

One consequence of initial algebra semantics, is that HITs are unique up to equality if the univalence axiom holds. This result is a consequence of the fact that the bicategory of algebras is *univalent*. Recall that a bicategory is univalent if equality between objects X and Y is equivalent to adjoint equivalences between X and Y and equality of 1-cells f and g is equivalent to invertible 2-cells between f and g [2]. Using the methods employed by Ahrens *et al.* one can show that the bicategory of algebras is univalent. Since biinitial objects are unique up to adjoint equivalence, one can conclude that HITs are unique up to equality.

Proposition 4.15. *Let Σ be a signature and let H_1 and H_2 are HITs for Σ. Denote the underlying algebras of H_1 and H_2 by X_1 and X_2. Then $X_1 = X_2$.*

5 Lifting the Groupoid Quotient

To construct higher inductive types, we use Proposition 4.14, which says that binitial objects satisfy the induction principle. We use the groupoid quotient to acquire the desired algebra. More specifically, we construct a pseudofunctor from $\mathrm{Alg}_{\mathrm{Grpd}}(\Sigma)$ to $\mathrm{Alg}(\Sigma)$, which is the groupoid quotient on the carrier. We do that in such a way that the obtained pseudofunctor preserves biinitiality, so that we obtain the HIT by constructing a biinitial object in $\mathrm{Alg}_{\mathrm{Grpd}}(\Sigma)$.

One class of pseudofunctors which preserve biinitial objects, are left biadjoints. Let us state that more precisely, so suppose that we have bicategories B and C, a left biadjoint pseudofunctor $L : \mathrm{Pseudo}(\mathsf{B}, \mathsf{C})$, and an object $x : \mathsf{B}$. Then the object $L(x)$ is biinitial if x is.

Instead of just lifting the groupoid quotient to the level of algebras, we first show that the groupoid quotient is a left biadjoint and then we lift that biadjunction to the level of algebras. To do so, we use the fact we defined the bicategory of algebras via displayed bicategories. This way we can define the biadjunction on each part of the structure separately.

More specifically, we define the notion of *displayed biadjunction* between two displayed bicategories over a biadjunction in the base, and we show that each displayed biadjunction gives rise to a total biadjunction between the total bicategories. Defining displayed biadjunctions requires defining displayed analogues of pseudofunctors, pseudotransformations, and invertible modification, which were defined by Ahrens *et al.* [2]. For this, we make use of *displayed invertible 2-cells* [2].

Definition 5.1. Let D_1 and D_2 be displayed bicategories over B_1 and B_2 respectively and let $F : \mathrm{Pseudo}(\mathsf{B}_1, \mathsf{B}_2)$ be a pseudofunctor. Then a **displayed pseudofunctor** $\overline{F}$ from D_1 to D_2 over F consist of

- For each $x : \mathsf{B}_1$ a map $\overline{F}_0 : \mathsf{D}_1(x) \to \mathsf{D}_2(F(x))$;
- For all 1-cells $f : x \to y$, objects $\overline{x} : \mathsf{D}_1(x)$ and $\overline{y} : \mathsf{D}_1(y)$, and displayed 1-cells $\overline{f} : \overline{x} \xrightarrow{f} \overline{y}$, a displayed 1-cell $\overline{F}_1(\overline{f}) : \overline{F}_0(\overline{x}) \xrightarrow{F(f)} \overline{F}_0(\overline{y})$;
- For all 2-cells $\theta : f \Rightarrow g$, displayed 1-cells $\overline{f} : \overline{x} \xrightarrow{f} \overline{y}$ and $\overline{g} : \overline{x} \xrightarrow{g} \overline{y}$, and displayed 2-cells $\overline{\theta} : \overline{f} \overset{\theta}{\Rightarrow} \overline{g}$, a displayed 2-cell $\overline{F}_2(\overline{\theta}) : \overline{F}_1(\overline{f}) \overset{F(\theta)}{\Longrightarrow} \overline{F}_2(\overline{g})$;
- For each $x : \mathsf{B}$ and $\overline{x} : \mathsf{D}(x)$, a displayed invertible 2-cell $\overline{F}_i(\overline{x}) : \mathrm{id}_1(\overline{F}_0(\overline{x})) \xrightarrow{F_i(x)} \overline{F}_1(\mathrm{id}_1(\overline{x}))$;
- For all $\overline{f} : \overline{x} \xrightarrow{f} \overline{y}$ and $\overline{g} : \overline{y} \xrightarrow{g} \overline{z}$. a displayed invertible 2-cell $\overline{F}_c(\overline{f}, \overline{g}) : \overline{F}_1(\overline{f}) \cdot \overline{F}_1(\overline{g}) \xrightarrow{F_c(f,g)} \overline{F}_1(\overline{f} \cdot \overline{g})$.

Here F_i and F_c denote the identitor and compositor of F. In addition, several coherencies, which can be found in the formalization, need to be hold. We denote the type of displayed pseudofunctors from D_1 to D_2 over F by $\mathsf{D}_1 \xrightarrow{F} \mathsf{D}_2$.

Definition 5.2. Let D_1 and D_2 be displayed bicategories over B_1 and B_2 respectively. Suppose that we have displayed pseudofunctors $\overline{F} : \mathsf{D}_1 \xrightarrow{F} \mathsf{D}_2$ and $\overline{G} : \mathsf{D}_1 \xrightarrow{G} \mathsf{D}_2$ and a

pseudotransformation $\theta : F \Rightarrow G$. Then a **displayed pseudotransformation** $\overline{\theta}$ from $\overline{F}$ to $\overline{G}$ over θ consists of

- For all objects $x : \mathsf{B}$ and $\overline{x} : \mathsf{D}_1(x)$ a displayed 1-cell $\overline{\theta}_1(x) : \overline{F}_0(\overline{x}) \xrightarrow{\theta(x)} \overline{G}_0(\overline{x})$;
- For all 1-cells $f : x \to y$ and $\overline{f} : \overline{x} \xrightarrow{f} \overline{y}$ a displayed invertible 2-cell $\overline{\theta}_2(f) : \overline{\theta}_1(\overline{x}) \cdot \overline{F}_1(\overline{f}) \overset{\theta(f)}{\Longrightarrow} \overline{G}_1(\overline{f}) \cdot \overline{\theta}_1(\overline{y})$.

Again several coherencies must be satisfied and the precise formulation can be found in the formalization. The type of displayed pseudotransformatons from $\overline{F}$ to $\overline{G}$ over θ is denoted by $\overline{F} \overset{\theta}{\Rightarrow} \overline{G}$.

Definition 5.3. Suppose that we have displayed bicategories D_1 and D_2 over B_1 and B_2, displayed pseudofunctors $\overline{F}$ and $\overline{G}$ from D_1 to D_2 over F and G respectively, and displayed pseudotransformations $\overline{\theta}$ and $\overline{\theta}'$ from $\overline{F}$ to $\overline{G}$ over θ and θ' respectively. In addition, let m be an invertible modification from θ to θ'. Then a **displayed invertible modification** $\overline{m}$ from $\overline{\theta}$ to $\overline{\theta}'$ over m consists of a displayed invertible 2-cell $\overline{m}_2(\overline{x}) : \overline{\theta}(\overline{x}) \overset{m(x)}{\Longrightarrow} \overline{\theta}'(\overline{x})$ for each $x : \mathsf{B}_1$ and $\overline{x} : \mathsf{D}_1(x)$, In addition, a coherency must be satisfied, which can be found in the formalization. The type of displayed invertible modifications from $\overline{\theta}$ to $\overline{\theta}'$ over m is denoted by $\overline{\theta} \overset{m}{\Longrightarrow} \overline{\theta}'$.

Each of these gadgets has a total version.

Problem 5.4. *We have*

1. *Given a displayed pseudofunctor* $\overline{F} : \mathsf{D}_1 \xrightarrow{F} \mathsf{D}_2$*, to construct a pseudofunctor* $\int \overline{F} : \mathsf{Pseudo}(\int \mathsf{D}_1, \int \mathsf{D}_2)$*;*
2. *Given a displayed pseudotransformation* $\overline{\theta} : \overline{F} \overset{\theta}{\Rightarrow} \overline{G}$*, to construct a pseudotransformation* $\int \overline{\theta} : \int \overline{F} \Rightarrow \int \overline{G}$*;*
3. *Given a displayed invertible modification* $\overline{m} : \overline{\theta} \overset{m}{\Rightarrow} \overline{\theta}'$*, to construct an invertible modificaton* $\int \overline{m} : \int \overline{\theta} \Rrightarrow \int \overline{\theta}'$*.*

Construction 5.5 (for Problem 5.4)**.** By pairing. □

Before we can define displayed biadjunctions, we need several operations on the displayed gadgets we introduced.

Example 5.6. We have the following

- We have $\mathrm{id}(\mathsf{D}) : \mathsf{D} \xrightarrow{\mathrm{id}(\mathsf{B})} \mathsf{D}$ where $\mathrm{id}(\mathsf{B})$ is the identity;
- Given $\overline{F} : \mathsf{D}_1 \xrightarrow{F} \mathsf{D}_2$ and $\overline{G} : \mathsf{D}_2 \xrightarrow{G} \mathsf{D}_3$, we have $\overline{F} \cdot \overline{G} : \mathsf{D}_1 \xrightarrow{F \cdot G} \mathsf{D}_3$ where $F \cdot G$ is the composition;
- Given $\overline{F} : \mathsf{D}_1 \xrightarrow{F} \mathsf{D}_2$, we have $\mathrm{id}_1(\overline{F}) : \overline{F} \overset{\mathrm{id}_1(F)}{\Longrightarrow} \overline{F}$ where $\mathrm{id}_1(F)$ is the identity transformation on F;
- Given $\overline{\theta} : \overline{F} \overset{\theta}{\Rightarrow} \overline{G}$ and $\overline{\theta}' : \overline{G} \overset{\theta'}{\Rightarrow} \overline{H}$, we have $\overline{\theta} \bullet \overline{\theta}' : \overline{F} \overset{\theta \bullet \theta'}{\Longrightarrow} \overline{H}$ where $\theta \bullet \theta'$ is the composition;
- Given $\overline{F} : \mathsf{D}_1 \xrightarrow{F} \mathsf{D}_2$, $\overline{G} : \mathsf{D}_2 \xrightarrow{G} \mathsf{D}_3$, $\overline{H} : \mathsf{D}_2 \xrightarrow{H} \mathsf{D}_3$, and $\overline{\theta} : \overline{G} \overset{\theta}{\Rightarrow} \overline{H}$, we have $\overline{F} \lhd \overline{\theta} : \overline{F} \cdot \overline{G} \xrightarrow{F \lhd \theta} \overline{F} \cdot \overline{H}$;
- Given $\overline{F} : \mathsf{D}_1 \xrightarrow{F} \mathsf{D}_2$, $\overline{G} : \mathsf{D}_1 \xrightarrow{G} \mathsf{D}_2$, $\overline{H} : \mathsf{D}_2 \xrightarrow{H} \mathsf{D}_3$, and $\overline{\theta} : \overline{F} \overset{\theta}{\Rightarrow} \overline{G}$, we have $\overline{\theta} \rhd \overline{H} : \overline{F} \cdot \overline{H} \xrightarrow{\theta \rhd H} \overline{G} \cdot \overline{H}$;
- Given $\overline{F} : \mathsf{D}_1 \xrightarrow{F} \mathsf{D}_2$, we have

$$\lambda : \mathrm{id} \cdot \overline{F} \overset{\lambda}{\Rightarrow} \overline{F}, \quad \rho : \overline{F} \cdot \mathrm{id} \overset{\rho}{\Rightarrow} \overline{F};$$

- Given $\overline{F} : \mathsf{D}_1 \xrightarrow{F} \mathsf{D}_2$, we have

$$\lambda^{-1} : \overline{F} \overset{\lambda^{-1}}{\Longrightarrow} \mathrm{id} \cdot \overline{F}, \quad \rho^{-1} : \overline{F} \overset{\rho^{-1}}{\Longrightarrow} \overline{F} \cdot \mathrm{id};$$

- Given $\overline{F} : \mathsf{D}_1 \xrightarrow{F} \mathsf{D}_2$, $\overline{G} : \mathsf{D}_2 \xrightarrow{G} \mathsf{D}_3$, and $\overline{H} : \mathsf{D}_3 \xrightarrow{H} \mathsf{D}_4$, we have

$$\alpha : (\overline{F} \cdot \overline{G}) \cdot \overline{H} \overset{\alpha}{\Rightarrow} \overline{F} \cdot (\overline{G} \cdot \overline{H}),$$

$$\alpha^{-1} : \overline{F} \cdot (\overline{G} \cdot \overline{H}) \overset{\alpha^{-1}}{\Longrightarrow} (\overline{F} \cdot \overline{G}) \cdot \overline{H}.$$

Definition 5.7. Suppose we have bicategories B_1 and B_2 and a biadjunction $L \dashv R$ from B_1 to B_2. We write η and ϵ for the unit and counit of $L \dashv R$ respectively, and we write τ_1 and τ_r for the left and right triangle respectively. Suppose, that we also have displayed bicategories D_1 and D_2 over B_1 and B_2 respectively and a displayed pseudofunctor $\overline{L} : \mathsf{D}_1 \xrightarrow{L} \mathsf{D}_2$. Then we say $\overline{L}$ is a **displayed left biadjoint pseudofunctor** if we have

- A displayed pseudofunctors $\overline{R} : \mathsf{D}_2 \xrightarrow{R} \mathsf{D}_1$;
- Displayed pseudotransformations

$$\overline{\eta} : \mathrm{id} \overset{\eta}{\Rightarrow} \overline{L} \cdot \overline{R}, \quad \overline{\epsilon} : \overline{R} \cdot \overline{L} \overset{\epsilon}{\Rightarrow} \mathrm{id};$$

- Displayed invertible modifications

$$\overline{\tau_1} : \ \rho^{-1} \bullet \overline{L} \lhd \overline{\eta} \bullet \alpha \bullet \overline{\epsilon} \rhd \overline{L} \bullet \lambda \overset{\tau_1}{\Longrightarrow} \mathrm{id}_1(\overline{L}),$$

$$\overline{\tau_2} : \ \lambda^{-1} \bullet \overline{\eta} \rhd \overline{R} \bullet \alpha^{-1} \bullet \overline{R} \lhd \overline{\epsilon} \bullet \rho \overset{\tau_2}{\Longrightarrow} \mathrm{id}_1(\overline{R}).$$

From Construction 5.5, we get

Proposition 5.8. *Given a displayed left biadjoint pseudofunctor* $\overline{L}$*, then* $\int \overline{L}$ *is a left biadjoint pseudofunctor.*

Now let us use the introduced notions to construct the biadjunction on the level of algebras. Our approach is summarized in Figure 4. We start by showing that the groupoid quotient gives rise to a biadjunction.

Problem 5.9. *To construct* $\mathsf{GQuot} \dashv \mathsf{PathGrpd}$ *with a pseudofunctor* $\mathsf{GQuot} : \mathsf{Pseudo}(\mathsf{Grpd}, \text{1-Type})$.

Construction 5.10 (for Problem 5.9)**.** We only show how the involved pseudofunctors are defined. The pseudofunctor GQuot is the groupoid quotient while PathGrpd sends a 1-type X to the groupoid whose objects are points of X and morphisms from x to y are paths $x = y$. □

Next we lift it to the level of algebras using the displayed machinery introduced in this section.

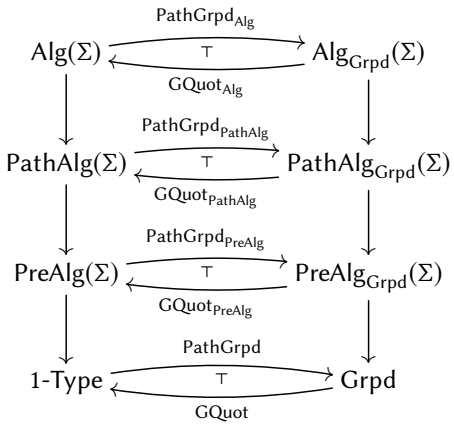

Figure 4. The biadjunction

Problem 5.11. *Given a signature* Σ*, to construct a biadjunction* $\mathsf{GQuot}_{\mathsf{Alg}} \dashv \mathsf{PathGrpd}_{\mathsf{Alg}}$ *where*

$$\mathsf{GQuot}_{\mathsf{Alg}} : \mathsf{Pseudo}(\mathsf{Alg}_{\mathsf{Grpd}}(\Sigma), \mathsf{Alg}(\Sigma)).$$

Construction 5.12 (for Problem 5.11)**.** We only give a very brief outline of the construction.

We start by constructing a displayed biadjunction from $\mathsf{DFalg}(\langle \mathsf{A}^{\Sigma} \rangle)$ to $\mathsf{DFalg}(\llbracket \mathsf{A}^{\Sigma} \rrbracket)$ over the biadjunction from Construction 5.10. To do so, we first need to lift the pseudofunctors, and for that, we generalize the approach of Hermida and Jacobs to the bicategorical setting [35, Theorem 2.14]. This requires us to construct two pseudotransformations.

$$p_1 : \llbracket P \rrbracket \cdot \mathsf{GQuot} \Rightarrow \mathsf{GQuot} \cdot \langle P \rangle,$$

$$p_2 : \langle P \rangle \cdot \mathsf{PathGrpd} \Rightarrow \mathsf{PathGrpd} \cdot \llbracket P \rrbracket.$$

We denote the total biadjunction of the resulting displayed biadjunction by $\mathsf{GQuot}_{\mathsf{PreAlg}} \dashv \mathsf{PathGrpd}_{\mathsf{PreAlg}}$.

Next we lift the biadjunction to the level of path algebras and for that, we construct a displayed biadjunction between $\mathsf{DFcell}(\langle \mathsf{l}^{\Sigma}(i) \rangle, \langle \mathsf{r}^{\Sigma}(i) \rangle)$ and $\mathsf{DFcell}(\llbracket \mathsf{l}^{\Sigma}(i) \rrbracket, \llbracket \mathsf{r}^{\Sigma}(i) \rrbracket$ for all $j : \mathsf{J}_{\mathsf{P}}$. Denote the resulting total biadjunction by $\mathsf{GQuot}_{\mathsf{PathAlg}} \dashv \mathsf{PathGrpd}_{\mathsf{PathAlg}}$.

To finish the proof, we need to construct one more displayed biadjunction. For that, we only need to show that if $G : \mathsf{PathAlg}_{\mathsf{Grpd}}(\Sigma)$ is an algebra, then $\mathsf{GQuot}_{\mathsf{PathAlg}}(G)$ also is an algebra, and if $X : \mathsf{PathAlg}(\Sigma)$ is an algebra, then so is $\mathsf{PathGrpd}_{\mathsf{PathAlg}}(X)$. □

The next proposition concludes this section.

Proposition 5.13. *If* G *is an biinitial object in* $\mathsf{Alg}_{\mathsf{Grpd}}(\Sigma)$*, then* $\mathsf{GQuot}_{\mathsf{Alg}}(G)$ *is a biinitial object in* $\mathsf{Alg}(\Sigma)$*.*

6 HIT Existence

From Theorem 4.14 we know that initiality implies the induction principle. Hence, it suffices to construct a biinitial object in the bicategory of algebras in 1-types. By Proposition 5.13, it suffices to construct a biinitial object in $\mathsf{Alg}_{\mathsf{Grpd}}(\Sigma)$. To do so, we adapt the semantics by Dybjer and Moeneclaey to our setting [28].

Problem 6.1. *Given a signature* Σ*, to construct a biinitial object* G *in* $\mathsf{Alg}_{\mathsf{Grpd}}(\Sigma)$*.*

Construction 6.2 (for Problem 6.1)**.** We only discuss how the carrier G of G is defined.

- Note that each polynomial P gives rise to a container $\hat{P}$. Note that each container induces a W-type [1], and we define the type of objects of G to be the W-type induced by $\hat{\mathsf{A}}$. Denote this type by G_0.
- The morphisms of G are constructed as a set quotient. We first define for each $x, y : G_0$ a type $x \sim y$ and for each $x, y : G_0$ and $f, g : x \sim y$, we define a type $f \approx g$. Both of these are defined as an inductive type and for the constructors, we refer the reader to the formalization. Basically, the constructors for these types are chosen in such a way that the groupoid being defined here, has the desired structure. This means we add constructors witnessing the path constructors, identity, composition, and all other laws. We use the same idea to define $f \approx g$.
  Note that the input of the quotient is an equivalence relation, which are valued in propositions. For this reason, we define $f \approx_p g$ to be the propositional truncation of $f \approx g$ All in all, we define the morphisms from x to y to be the set quotient of $x \sim y$ by $\approx_p$. □

Problem 6.3. *Each signature has a HIT.*

Construction 6.4 (for Problem 6.3)**.** By Propositions 4.14 and 5.13, it suffices to find a biinitial object in $\mathsf{Alg}_{\mathsf{Grpd}}(\Sigma)$. The desired object is given in Construction 6.2. □

7 Conclusion and Further Work

We showed how to construct finitary 1-truncated higher inductive types using the propositional truncation, quotient, and the groupoid quotient. This reduces the existence of a general class of HITs to simpler ones. We needed the types to be 1-truncated, so that we could use the framework of bicategory theory, and the HITs we studied had to be finitary to guarantee that the groupoid quotient commutes with the involved operations [23]. On the way, we also proved that HITs are unique.

There are numerous ways to improve on this result. First of all, the bicategory of algebras in 1-types can be studied in more detail. For example, it should have products, inserters and equifiers [55]. In addition, using the fact that we have higher inductive types, we should be able to show that this bicategory also has coproducts, co-inserters, and co-equifiers. Furthermore, to connect our approach to algebra in bicategories with established approaches, one should show that the underlying functor from $\mathsf{Alg}(\Sigma)$ to 1-Type has a left biadjoint, which gives rise to a monad on 1-Type [19, 44]. The

biadjunction could be constructed using higher inductive types.

Second of all, one would like to get rid of the truncation level. Since untruncated types correspond to ∞-groupoids, generalizing the methods used in this paper to the untruncated case, requires formalizing notions from ∞-category theory in type theory [9, 21, 30]. This also requires finding an ∞-dimensional generalization of the groupoid quotient. An alternative approach to deal with untruncated HITs, pointed out by Ali Caglayan, would be using wild categories [36, 43].

Lastly, it should be possible to take advantage of the way we constructed HITs to say something about the path space. One can show with the encode-decode method that the type $\mathbf{gcl}(x) = \mathbf{gcl}(y)$ is equivalent to $G(x, y)$. By inspecting Construction 6.4, we see that HITs are constructed as the groupoid quotient of the groupoid G constructed in Construction 6.2 where we also proved a universal property for G. For concrete examples, such as the circle, one might be able to make use of this universal property to deduce a mapping principle for $\mathbf{base}_{S^1} = \mathbf{base}_{S^1}$, which could be used to show that $\pi_1(S^1)$ is the integers [49].

Acknowledgments

The author thanks Herman Geuvers, Dan Frumin, Niccolò Veltri, Benedikt Ahrens, and Ali Caglayan for helpful comments and discussions. The author also thanks the anonymous reviewers for their helpful comments and suggestions.

References

[1] Michael Abbott, Thorsten Altenkirch, and Neil Ghani. 2003. Categories of Containers. In *International Conference on Foundations of Software Science and Computation Structures*. Springer, 23–38. https://doi.org/10.1007/3-540-36576-1_2

[2] Benedikt Ahrens, Dan Frumin, Marco Maggesi, Niccoló Veltri, and Niels van der Weide. 2020. Bicategories in Univalent Foundations. *arXiv preprint arXiv:1903.01152v3v2* (2020). https://arxiv.org/abs/1903.01152v2

[3] Benedikt Ahrens, Krzysztof Kapulkin, and Michael Shulman. 2015. Univalent categories and the Rezk completion. *Mathematical Structures in Computer Science* 25 (2015), 1010–1039. https://doi.org/10.1017/S0960129514000486 arXiv:1303.0584

[4] Benedikt Ahrens and Peter LeFanu Lumsdaine. 2019. Displayed Categories. *Logical Methods in Computer Science* 15, 1 (2019). https://doi.org/10.23638/LMCS-15(1:20)2019

[5] Thorsten Altenkirch, Paolo Capriotti, Gabe Dijkstra, Nicolai Kraus, and Fredrik Nordvall Forsberg. 2018. 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*. 293–310. https://doi.org/10.1007/978-3-319-89366-2_16

[6] Thorsten Altenkirch, Nils Anders Danielsson, and Nicolai Kraus. 2017. 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*. 534–549. https://doi.org/10.1007/978-3-662-54458-7_31

[7] Thorsten Altenkirch and Ambrus Kaposi. 2016. 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*. 18–29. https://doi.org/10.1145/2837614.2837638

[8] Thorsten Altenkirch and Ambrus Kaposi. 2017. Normalisation by Evaluation for Type Theory, in Type Theory. *Logical Methods in Computer Science* 13, 4 (2017). https://doi.org/10.23638/LMCS-13(4:1)2017

[9] Thorsten Altenkirch and Ondrej Rypacek. 2012. 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*. 16–30. https://doi.org/10.4230/LIPIcs.CSL.2012.16

[10] Thorsten Altenkirch and Luis Scoccola. 2019. The Integers as a Higher Inductive Types. http://luisscoccola.github.io/Luis%20Scoccola_files/int-as-hit.pdf

[11] Carlo Angiuli, Robert Harper, and Todd Wilson. 2017. 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*. 680–693. http://dl.acm.org/citation.cfm?id=3009861

[12] Carlo Angiuli, Edward Morehouse, Daniel R. Licata, and Robert Harper. 2016. Homotopical Patch Theory. *J. Funct. Program.* 26 (2016), e18. https://doi.org/10.1017/S0956796816000198

[13] 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*. ACM, 76–85. https://doi.org/10.1145/3209108.3209130

[14] Steven Awodey, Nicola Gambino, and Kristina Sojakova. 2012. 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*. 95–104. https://doi.org/10.1109/LICS.2012.21

[15] Steve Awodey and Michael A Warren. 2009. Homotopy theoretic models of identity types. In *Mathematical proceedings of the cambridge philosophical society*, Vol. 146. Cambridge University Press, 45–55.

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

[17] Jean Bénabou. 1967. Introduction to bicategories. In *Reports of the Midwest Category Seminar*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1–77. https://doi.org/10.1007/BFb0074299

[18] Marc Bezem, Thierry Coquand, and Simon Huber. 2013. 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*. 107–128. https://doi.org/10.4230/LIPIcs.TYPES.2013.107

[19] Robert Blackwell, Gregory M Kelly, and John Power. 1989. Two-dimensional monad theory. *Journal of Pure and Applied Algebra* 59, 1 (1989), 1–41. https://doi.org/10.1016/0022-4049(89)90160-6

[20] Ulrik Buchholtz and Egbert Rijke. 2017. The real projective spaces in homotopy type theory. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017*. 1–8. https://doi.org/10.1109/LICS.2017.8005146

[21] Paolo Capriotti and Nicolai Kraus. 2018. Univalent higher categories via complete Semi-Segal types. *PACMPL* 2, POPL (2018), 44:1–44:29. https://doi.org/10.1145/3158132

[22] Evan Cavallo and Robert Harper. 2019. Higher Inductive Types in Cubical Computational Type Theory. *PACMPL* 3, POPL (2019), 1:1–1:27. https://doi.org/10.1145/3290314

[23] James Chapman, Tarmo Uustalu, and Niccolò Veltri. 2019. Quotienting the Delay Monad by Weak Bisimilarity. *Mathematical Structures in Computer Science* 29, 1 (2019), 67–92. https://doi.org/10.1017/S0960129517000184

[24] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg. 2017. Cubical Type Theory: A Constructive Interpretation of the

Univalence Axiom. *FLAP* 4, 10 (2017), 3127–3170.

[25] Thierry Coquand, Simon Huber, and Anders Mörtberg. 2018. 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*. 255–264. https://doi.org/10.1145/3209108.3209197

[26] Thierry Coquand, Simon Huber, and Christian Sattler. 2019. Homotopy Canonicity for Cubical Type Theory. In *4th International Conference on Formal Structures for Computation and Deduction, FSCD 2019, June 24-30, 2019, Dortmund, Germany*. 11:1–11:23. https://doi.org/10.4230/LIPIcs.FSCD.2019.11

[27] Peter Dybjer. 1994. Inductive Families. *Formal aspects of computing* 6, 4 (1994), 440–465. https://doi.org/10.1007/BF01211308

[28] Peter Dybjer and Hugo Moeneclaey. 2018. Finitary Higher Inductive Types in the Groupoid Model. *Electr. Notes Theor. Comput. Sci.* 336 (2018), 119–134. https://doi.org/10.1016/j.entcs.2018.03.019

[29] Peter Dybjer and Anton Setzer. 1999. A Finite Axiomatization of Inductive-Recursive Definitions. In *TLCA*. https://doi.org/10.1007/3-540-48959-2_11

[30] Eric Finster and Samuel Mimram. 2017. 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*. 1–12. https://doi.org/10.1109/LICS.2017.8005124

[31] Fredrik Nordvall Forsberg and Anton Setzer. 2010. Inductive-Inductive Definitions. In *International Workshop on Computer Science Logic*. Springer, 454–468. https://doi.org/10.1007/978-3-642-15205-4_35

[32] Fredrik Nordvall Forsberg and Anton Setzer. 2012. A Finite Axiomatisation of Inductive-Inductive Definitions. *Logic, Construction, Computation* 3 (2012), 259–287.

[33] 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*. ACM, 201–214. https://doi.org/10.1145/3167085

[34] Nick Gurski. 2012. Biequivalences in tricategories. *Theory and Applications of Categories* 26, 14 (2012), 349–384.

[35] Claudio Hermida and Bart Jacobs. 1998. Structural Induction and Coinduction in a Fibrational Setting. *Information and computation* 145, 2 (1998), 107–152. https://doi.org/10.1006/inco.1998.2725

[36] André Hirschowitz, Tom Hirschowitz, and Nicolas Tabareau. 2015. 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*. 226–240. https://doi.org/10.4230/LIPIcs.TLCA.2015.226

[37] Martin Hofmann and Thomas Streicher. 1994. 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*. 208–212. https://doi.org/10.1109/LICS.1994.316071

[38] Martin Hofmann and Thomas Streicher. 1998. The groupoid interpretation of type theory. In *Twenty-five years of constructive type theory (Venice, 1995)*. Oxford Logic Guides, Vol. 36. Oxford Univ. Press, New York, 83–111.

[39] Ambrus Kaposi and András Kovács. 2018. 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*. 20:1–20:18. https://doi.org/10.4230/LIPIcs.FSCD.2018.20

[40] Ambrus Kaposi, András Kovács, and Thorsten Altenkirch. 2019. Constructing Quotient Inductive-Inductive Types. *PACMPL* 3, POPL (2019), 2:1–2:24. https://doi.org/10.1145/3290315

[41] Chris Kapulkin and Peter LeFanu Lumsdaine. 2018. The Simplicial Model of Univalent Foundations (after Voevodsky). *Journal of the European Mathematical Society* (2018). arXiv:1211.2851

[42] 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, New York, NY, USA, July 5-8, 2016*. 595–604. https://doi.org/10.1145/2933575.2933586

[43] Nicolai Kraus and Jakob von Raumer. 2019. Path Spaces of Higher Inductive Types in Homotopy Type Theory. In *34th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2019, Vancouver, BC, Canada, June 24-27, 2019*. 1–13. https://doi.org/10.1109/LICS.2019.8785661

[44] Stephen Lack. 2000. A Coherent Approach to Pseudomonads. *Advances in Mathematics* 152, 2 (2000), 179 – 202.

[45] Tom Leinster. 1998. Basic Bicategories. arXiv:math/9810017

[46] Daniel R Licata and Guillaume Brunerie. 2013. $\pi_n(S^n)$ in Homotopy Type Theory. In *International Conference on Certified Programs and Proofs*. Springer, 1–16. https://doi.org/10.1007/978-3-319-03545-1_1

[47] Daniel R Licata and Guillaume Brunerie. 2015. A Cubical Approach to Synthetic Homotopy Theory. In *2015 30th Annual ACM/IEEE Symposium on Logic in Computer Science*. IEEE, 92–103. https://doi.org/10.1109/LICS.2015.19

[48] Daniel R. Licata and Eric Finster. 2014. 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*. 66:1–66:9. https://doi.org/10.1145/2603088.2603153

[49] Daniel R. Licata and Michael Shulman. 2013. 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*. 223–232. https://doi.org/10.1109/LICS.2013.28

[50] Peter LeFanu Lumsdaine. 2010. Weak ω-categories from Intensional Type Theory. *Logical Methods in Computer Science* 6, 3. https://doi.org/10.2168/LMCS-6(3:24)2010

[51] Peter LeFanu Lumsdaine and Michael Shulman. 2017. Semantics of higher inductive types. *Mathematical Proceedings of the Cambridge Philosophical Society* (2017), 1–50. https://doi.org/10.1017/S030500411900015X

[52] Per Martin-Löf. 1975. An Intuitionistic Theory of Types: Predicative Part. In *Studies in Logic and the Foundations of Mathematics*. Vol. 80. Elsevier, 73–118.

[53] The Coq development team. 2018. *The Coq Proof Assistant*. Version 8.8.

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

[55] John Power and Edmund Robinson. 1991. A characterization of pie limits. In *Mathematical Proceedings of the Cambridge Philosophical Society*, Vol. 110. Cambridge University Press, 33–47.

[56] Egbert Rijke. 2017. The Join Construction. *arXiv preprint arXiv:1701.07538* (2017).

[57] 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 2015, Mumbai, India, January 15-17, 2015*. 31–42. https://doi.org/10.1145/2676726.2676983

[58] Kristina Sojakova. 2016. Higher Inductive Types as Homotopy-Initial Algebras. http://reports-archive.adm.cs.cmu.edu/anon/2016/CMU-CS-16-125.pdf

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

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

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

2019.09.014

[62] 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, Saint Petersburg, FL, USA, January 20-22, 2016*. 122–129. https://doi.org/10.1145/2854065.2854076

[63] Andrea Vezzosi, Anders Mörtberg, and Andreas Abel. 2019. Cubical Agda: a dependently typed programming language with univalence and higher inductive types. *Proceedings of the ACM on Programming Languages* 3, ICFP (2019), 1–29. https://doi.org/10.1145/3341691

[64] Vladimir Voevodsky, Benedikt Ahrens, Daniel Grayson, et al. 2020. UniMath — a computer-checked library of univalent mathematics. Available at https://github.com/UniMath/UniMath. Accessed on 13 December 2019, `git` hash 2dadfb61.