A mechanized study of coherent 2-groups

Perry Hart

University of Minnesota, Twin Cities

May 14, 2025

Contents

1 Introduction

Working in Book HoTT, we construct a fully verified biequivalence between the $(2,1)$-category of coherent 2-groups [2] and the $(2,1)$-category of pointed (0-)connected 2-types. We also obtain an equality between them as a corollary. This biequivalence has been suggested at a few places in the literature. Inside HoTT, [4, Section 9] proposes it as a 2-dimensional generalization of the equivalence the authors construct between **Grp** and pointed connected 1-types. It also was suggested in the classical setting by [2, Section 8.2].

Indeed, the biequivalence we construct generalizes the 1-dimensional equivalence. It consists of two broad steps. First, we construct the delooping of a coherent 2-group G as a HIT generalizing the first Eilenberg-MacLane space [9]. This defines a function from the type of coherent 2-groups to the type of pointed connected 2-types. Second, we equip this function with the structure of a pseudofunctor and prove that it forms a biequivalence [1, Definition 2.17] with the loop space pseudofunctor. Each step is purely algebraic but involves several huge computations.

In the rest of this paper, we review basic notions of bicategory theory while focusing on the $(2,1)$-category of coherent 2-groups and the $(2,1)$-category of pointed connected 2-types (Section 2).

Afterward, we outline the major computations involved in the two steps of the biequivalence (Sections 3 and 4). Finally, we deduce from the biequivalence an identity between the $(2,1)$-categories in question via univalence and a bit of wild category theory (Section 5). This outline also will serve as a roadmap for our Agda codebase [7], which has the complete biequivalence and the identity we derive from it.

2 Bicategories

In this work, *bicategory* means $(2,1)$ category whose 2-cells are paths. This definition is a special case of the traditional one, in which 2-cells are simply elements of a family of sets [1, Definition 2.1]. In particular, the theory of [1] applies to our theory of bicategories. Our theory is noticeably simpler since the identity type already carries much of the data and satisfies many of the properties required of 2-cells.

Definition 2.0.1 ([7, BicatStr]). Let $\mathcal{U}_1$ and $\mathcal{U}_2$ be universes. A *bicategory (relative to $\mathcal{U}_1$ and $\mathcal{U}_2$)* consists of a type $\mathsf{Ob} : \mathcal{U}_1$ of objects together with

- a doubly indexed family hom of 1-types in $\mathcal{U}_2$ over Ob, whose elements are called *morphisms* or 1-*cells*
- a composition operation $\circ : \mathsf{hom}(b,c) \to \mathsf{hom}(a,b) \to \mathsf{hom}(a,c)$ for all $a, b, c : \mathsf{Ob}$
- an identity morphism id_a for each $a : \mathsf{Ob}$ together with two 2-cells, called the *right unitor* and *left unitor*, witnessing that each identity morphism is a left unit and a right unit, respectively, for $\circ$.
- a 2-cell, called the *associator*, witnessing that $\circ$ is associative and satisfying both the triangle identity with the unitors and the pentagon identity.

Remark 2.0.2. The structure of a bicategory on Ob in the sense of Definition 2.0.1 is equivalent to the structure of a *locally univalent bicategory* in the sense of [1, Definition 3.1] where all 2-cells are invertible.

Definition 2.0.3 ([7, AdjEq]). Let $\mathcal{C}$ be a bicategory. Let $a, b : \mathsf{Ob}(\mathcal{C})$ and $f : \mathsf{hom}_{\mathcal{C}}(a,b)$. We say that f is an *adjoint equivalence* if we have a morphism $g : \mathsf{hom}_{\mathcal{C}}(b,a)$, 2-cells $\eta : g \circ f = \mathsf{id}_a$ and $\epsilon : f \circ g = \mathsf{id}_b$, and two triangle-like identities. We denote the type of adjoint equivalences between a and b by $\mathsf{AdjEquiv}(a,b)$.

Example 2.0.4. We have the bicategory $\mathbf{2Type}_{\mathbf{0}}^{*}$ of pointed connected 2-types and pointed maps [7, Ptd-bc]. The fact that each of its hom-types is 1-truncated follows from [4, Corollary 4.3], which we have mechanized at [7, PtdFibration].

Example 2.0.5. We have the bicategory **2Grp** of *(coherent) 2-groups* and 2-group morphisms. A 2-group is a monoidal category where, from the viewpoint of a monoidal category as a single-object bicategory, every object is equipped with an adjoint equivalence. Explicitly, given a universe $\mathcal{U}$, a 2-*group relative to* $\mathcal{U}$ [7, CohGrp] is a 1-type G in $\mathcal{U}$ equipped with

- a basepoint id
- a binary operation $\otimes : G \to G \to G$, called the *tensor product*
- a right unitor ρ, a left unitor λ, and an associator α for $\otimes$
- a triangle identity and a pentagon identity
- an *inverse* operation $(-)^{-1} : G \to G$
- paths $\mathsf{linv}_x : x^{-1} \otimes x = \mathsf{id}$ and $\mathsf{rinv}_x : x \otimes x^{-1} = \mathsf{id}$ for each $x : G$ such that linv and rinv satisfy two zig-zag identities.

Remark 2.0.6. A 2-group is the same as a single-object bicategory where every 1-cell x is an adjoint equivalence. Also, it is the same as a *coherent* 2-*group object* [2, Definition 7.1] in the bicategory of 1-types.

A 2-*group morphism* $G_1 \to G_2$ is a function $f_0 : G_1 \to G_2$ equipped with a family of paths $\mu_{x,y} : f_0(x) \otimes f_0(y) = f_0(x \otimes y)$ that respects the associator [7, CohGrpHomStr].

Note 2.0.7. Our notion of 2-*group morphism* is surprisingly short: a morphism of the underlying coherent semigroups. The correct notion must preserve *all* data of a 2-group, not just the tensor product and the associator [7, CohGrpHomStrFull]. To justify the short definition, we prove that for each function $f_0 : G_1 \to G_2$ between the underlying types of 2-groups, the forgetful function

$$\textit{fully explicit notion on } f_0 \quad \to \quad \textit{short notion on } f_0 \qquad (\mathtt{ForgMap}(f_0))$$

is an equivalence [7, 2GrpHomEq]. We include the de-formalized proof of this equivalence in Section A. The short definition is highly valuable as it lets us define the classifying space of a 2-group G as a HIT $K_2(G)$ with fewer constructors (Section 3), thereby making induction on $K_2(G)$ much simpler.

Note 2.0.8. By the structure identity principle (SIP) [11, The structure identity principle], 2-cells between 2-group morphisms $f, g : G_1 \to G_2$ are equivalent to *natural isomorphisms* between f and g. A *natural isomorphism* is a homotopy $\mathsf{fun}(f) \sim \mathsf{fun}(g)$ between the underlying functions that commutes with the tensor product [7, 2Grp]. For example, we build the unitors and associator for $\mathsf{2Grp}$ via natural isomorphisms [7, 2SGrpMap].

Example 2.0.9 ([7, Hmtpy2Grp]). For every pointed 2-type X, the loop space $\Omega(X)$ equipped with path composiiton has the structure of a 2-group, called the *fundamental* 2-*group* of X. Also, for each function $f : X \to_* Y$ between pointed 2-types, we have a morphism $\Omega(f) : \Omega(X) \to \Omega(Y)$ of 2-groups. This action on morphisms preserves both the identity map and composition of maps.

Example 2.0.10 ([7, PostMultMap]). Let G be a 2-group and $g : G$. The function $\mathsf{post\text{-}mult}_g : G \to G$ defined by $x \mapsto x \otimes g$ is a 2-group morphism.

Example 2.0.11. Let X be a type. The type $X \simeq X$ of self-equivalences is a coherent semigroup [7, $\simeq$-2SGrp]. The function $\mathsf{univ}_X : (X \simeq X) \to (X = X)$ is a morphism of coherent semigroups [7, ua-2SGrpMap].

We end this section by providing, via the SIP, sufficient conditions for morphisms to be adjoint equivalences in the two bicategories we care about. Note that Note 2.0.7 is essential for deriving this result for **2Grp**.

Lemma 2.0.12 ([7, AdjEq-exmps]).

(1) For every morphism f in $\mathbf{2Type}_0^*$, if its underlying function $\mathsf{fun}(f)$ is an equivalence of types, then it is an adjoint equivalence.

*(2) For every morphism f in **2Grp**, if its underlying function $\mathsf{fun}(f)$ is an equivalence of types, then it is an adjoint equivalence.*

3 Delooping a 2-group

The first Eilenberg-MacLane space of a group H, also known as the *classifying space of H*, is defined as the 1-truncated HIT $K(H,1)$ generated by $\mathsf{base} : K(H,1)$ and $\mathsf{loop} : H \to \mathsf{base} = \mathsf{base}$ along with a higher-path constructor $\mathsf{loop\text{-}comp}$ witnessing that loop is a group morphism $H \to \Omega(K(H,1))$. Let $\mathcal{U}$ be a universe and G be a 2-group relative to $\mathcal{U}$. We define the *classifying space of a 2-group G* as the 2-truncated HIT $K_2(G)$ generated by $\mathsf{base} : K_2(G)$ and $\mathsf{loop} : G \to \mathsf{base} = \mathsf{base}$ along with two higher-path constructors $\mathsf{loop\text{-}comp}$ and $\mathsf{loop\text{-}assoc}$ witnessing that loop is a 2-group morphism $G \to \Omega(K_2(G))$ [7, Delooping]. To state the induction principle, we need *higher* dependent paths for the input data corresponding to $\mathsf{loop\text{-}comp}$ and $\mathsf{loop\text{-}assoc}$. Such notions are defined by path induction, and the definitions we choose let us visualize higher dependent paths as fillers of hollow 2-dimensional and 3-dimensional cylinders [7, PathPathOver]. The recursion principle, derived from the induction principle, states that $K_2(G)$ is initial in the wild category of pointed 2-types X^* equipped with a 2-group morphism $G \to \Omega(X^*)$. Explicitly, for every pointed 2-type $X^* :\equiv (X, x_0)$ together with a 2-group morphism $\varphi_{X^*} : G \to \Omega(X^*)$, we have a function $M_\varphi : K_2(G) \to X$ that satisfies $M_\varphi(\mathsf{base}) \equiv x_0$ and is equipped with a natural isomorphism

$$\begin{tikzcd} & G \arrow[dl, "\mathsf{loop}"'] \arrow[dr, "\varphi_{X^*}"] \arrow[d, phantom, "(\rho_\varphi, \tilde{\rho}_\varphi)"] & \\ \Omega(K_2(G)) \arrow[rr, "\Omega(M_\varphi)"'] & {} & \Omega(X^*) \end{tikzcd}$$

of 2-group morphisms. We call ρ_φ the *point computation rule* and $\tilde{\rho}_\varphi$ the *tensor computation rule.*

Lemma 3.0.1 ([7, K_2-is-conn]). *The type $K_2(G)$ is connected.*

A fundamental property of $K(H,1)$ is that it is the *delooping* of H, i.e., that loop is a group isomorphism. We want to show that, similarly, $K_2(G)$ is the delooping of G. By Lemma 2.0.12(2), it suffices to show that loop is an equivalence of types. We adapt the encode-decode proof used for $K(H,1)$ [9, Theorem 3.2] to our higher-dimensional setting.

We define $\mathsf{codes} : K_2(G) \to \mathcal{U}_{\leq 1}$ by recursion on $K_2(G)$ so that $\mathsf{pr}_1(\mathsf{codes}(\mathsf{base})) \equiv G$ [7, codes], where $\mathcal{U}_{\leq 1}$ denotes the type of all 1-truncated types in $\mathcal{U}$. Since G is 1-truncated by definition, we may take it as the basepoint of $\mathcal{U}_{\leq 1}$. To construct codes, it suffices to construct a 2-group morphism $\zeta : G \to \Omega(\mathcal{U}_{\leq 1}, G)$. Define $\zeta_{\mathsf{map}} : G \to (G = G)$ by mapping g to the equivalence

$$\begin{aligned} \mathsf{post\text{-}mult}_g \ &: \ G \xrightarrow{\simeq} G \\ \mathsf{post\text{-}mult}_g(x) \ &:= \ x \otimes g \end{aligned}$$

and then applying univ to $\mathsf{post\text{-}mult}_g$. Both $\mathsf{post\text{-}mult}$ and univ are morphisms of coherent semigroups (Examples 2.0.10 and 2.0.11, respectively), and we give ζ_{map} the composite of their morphism structures. Now, let $\mathsf{codes}_0 := \mathsf{pr}_1 \circ \mathsf{codes}$ and define

$$\begin{aligned} \mathsf{encode} \ &: \ \prod_{z : K_2(G)} \mathsf{base} = z \to \mathsf{codes}_0(z) \\ \mathsf{encode}(z, p) \ &:= \ \mathsf{transp}^{\mathsf{codes}_0}(p, \mathsf{e}_G) \end{aligned}$$

This gives us a function $\mathsf{encode}(\mathsf{base}) : \Omega(K_2(G)) \to G$ [7, encode].

We want to show that $\mathsf{loop} : G \to \Omega(K_2(G))$ is an equivalence with inverse $\mathsf{encode}(\mathsf{base})$. As in [9], $\mathsf{encode}(\mathsf{base})$ is a left inverse of loop, mechanized in [7, encode-loop]. The main ingredient for the proof of this is the chain of paths

$$\begin{gathered} \mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(x), y) \\ \| \\ \textit{via path induction on } \mathsf{loop}(x) \\ \Downarrow \\ \mathsf{coe}(\mathsf{ap}_{\mathsf{pr}_1}(\mathsf{ap}_{\mathsf{codes}}(\mathsf{loop}(x))), y) \\ \| \\ \textit{via } \mathsf{codes}\textit{'s point computation rule} \\ \Downarrow \\ \mathsf{coe}(\zeta_{\mathsf{map}}(x), y) \\ \| \\ \textit{univalence axiom} \\ \Downarrow \\ y \otimes x \end{gathered}$$

for all $x, y : G$, denoted by $\mathsf{transp\text{-}codes}(x, y)$. This term also plays an important role in the next part of the proof, for which we record the following coherence property.

Lemma 3.0.2 ([7, coe-β-mu]). *For all $x, y, z : G$, the following diagram commutes.*

$$\begin{tikzcd}
\mathsf{coe}(\mathsf{univ}(\mathsf{post\text{-}mult}(x \otimes y)), z) \arrow[r, equal, "\textit{univalence axiom at } \mathsf{post\text{-}mult}(x \otimes y)"] \arrow[d, equal, "\textit{associativity of } \otimes"'] & z \otimes (x \otimes y) \arrow[d, equal, "\textit{associativity of } \otimes"] \\
\mathsf{coe}(\mathsf{univ}(\mathsf{post\text{-}mult}(y) \circ \mathsf{post\text{-}mult}(x)), z) \arrow[dd, equal, "\mathsf{univ} \textit{ respects composition}"'] & (z \otimes x) \otimes y \arrow[d, equal, "\textit{univalence axiom at } \mathsf{post\text{-}mult}(y)"] \\
 & \mathsf{coe}(\mathsf{univ}(\mathsf{post\text{-}mult}(y)), z \otimes x) \arrow[d, equal, "\textit{univalence axiom at } \mathsf{post\text{-}mult}(x)"] \\
\mathsf{coe}(\mathsf{univ}(\mathsf{post\text{-}mult}(x)) \cdot \mathsf{univ}(\mathsf{post\text{-}mult}(y)), z) \arrow[r, equal, "\mathsf{coe} \textit{ respects composition}"'] & \mathsf{coe}(\mathsf{univ}(\mathsf{post\text{-}mult}(y)), \mathsf{coe}(\mathsf{univ}(\mathsf{post\text{-}mult}(x)), z))
\end{tikzcd}$$

Next, we show that $\mathsf{encode}(\mathsf{base})$ is a right inverse of loop. We want a homotopy $\eta : \mathsf{loop} \circ \mathsf{encode}(\mathsf{base}) \sim \mathsf{id}_{\Omega(K_2(G))}$. To this end, we will define

$$\mathsf{decode} \ : \prod_{z : K_2(G)} \mathsf{codes}_0(z) \to \mathsf{base} = z$$

by induction on $K_2(G)$ so that $\mathsf{decode}(\mathsf{base}) \equiv \mathsf{loop}$. By path induction, it then follows that $\mathsf{decode}_z(\mathsf{encode}_z(p)) = p$ for all $z : K_2(G)$ and $p : \mathsf{base} = z$ because every 2-group morphism, such as loop, preserves the identity. This gives us η, as desired.

We now describe the construction of decode [7, Decode-def], which is much more complex than the 1-dimensional case. Here, the target of the induction is the function type $\mathsf{codes}_0(z) \to \mathsf{base} = z$ for all $z : K_2(G)$. In such a situation, we have the following form of the induction principle, which is useful for computations.

Lemma 3.0.3 ([7, PPOverFun]). *Let B_1 be a type family over $K_2(G)$ and B_2 a family of 1-types over $K_2(G)$. Suppose we have a function $\psi_{\mathsf{base}} : B_1(\mathsf{base}) \to B_2(\mathsf{base})$ together with*

- *for each $x : G$, a function $\psi_{\mathsf{loop}}(x) : \prod_{b : B_1(\mathsf{base})} \psi_{\mathsf{base}}(\mathsf{transp}^{B_1}(\mathsf{loop}(x), b)) = \mathsf{transp}^{B_2}(\mathsf{loop}(x), \psi_{\mathsf{base}}(b))$*
- *for all $x, y : G$ and $b : B_1(\mathsf{base})$, a commuting diagram of paths*

$$\begin{tikzcd}[column sep=small]
 & \psi_{\mathsf{base}}(\mathsf{transp}^{B_1}(\mathsf{loop}(x \otimes y), b)) \arrow[dl, equal, "\textit{via } \mathsf{loop\text{-}comp}(x,y)"'] \arrow[dr, equal, "\psi_{\mathsf{loop}}(x \otimes y, b)"] & \\
\psi_{\mathsf{base}}(\mathsf{transp}^{B_1}(\mathsf{loop}(x) \cdot \mathsf{loop}(y), b)) \arrow[d, equal, "\mathsf{transp} \textit{ respects path composition}"'] & & \mathsf{transp}^{B_2}(\mathsf{loop}(x \otimes y), \psi_{\mathsf{base}}(b)) \arrow[d, equal, "\textit{via } \mathsf{loop\text{-}comp}(x,y)"] \\
\psi_{\mathsf{base}}(\mathsf{transp}^{B_1}(\mathsf{loop}(y), \mathsf{transp}^{B_1}(\mathsf{loop}(x), b))) \arrow[d, equal, "\psi_{\mathsf{loop}}(y, \mathsf{transp}^{B_1}(\mathsf{loop}(x), \mathsf{transp}^{B_1}(\mathsf{loop}(x), b)))"'] & \boxed{\psi_{\mathsf{loop\text{-}comp}}(x, y, b)} & \mathsf{transp}^{B_2}(\mathsf{loop}(x) \cdot \mathsf{loop}(y), \psi_{\mathsf{base}}(b)) \arrow[d, equal, "\mathsf{transp} \textit{ respects path composition}"] \\
\mathsf{transp}^{B_2}(\mathsf{loop}(y), \psi_{\mathsf{base}}(b)) \arrow[rr, equal, "\textit{via } \psi_{\mathsf{loop}}(x, b)"'] & & \mathsf{transp}^{B_2}(\mathsf{loop}(y), \mathsf{transp}^{B_2}(\mathsf{loop}(x), \psi_{\mathsf{base}}(b)))
\end{tikzcd}$$

Then we have a function $\psi : \prod_{x:K_2(G)} B_1(x) \to B_2(x)$ *that satisfies* $\psi(\mathsf{base}) \equiv \psi_{\mathsf{base}}$.

Note that Lemma 3.0.3 avoids the input data for $\mathsf{loop\text{-}assoc}$ because the target of the induction is a 1-type. By instantiating B_1 with $\mathsf{codes}_0(z)$ and B_2 with $\mathsf{base} = z$, Lemma 3.0.3 gives us a sufficient condition for constructing decode, namely the data ψ_{base}, ψ_{loop}, and $\psi_{\mathsf{loop\text{-}comp}}$. Of course, we define ψ_{base} as loop. For all $x, y : G$, we define $\psi_{\mathsf{loop}}(x, y)$ as the chain of paths

$$\begin{gathered}
\mathsf{loop}(\mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(x), y)) \\
\| \\
{\scriptstyle \mathsf{ap}_{\mathsf{loop}}(\mathsf{transp\text{-}codes}(x,y))} \\
\Downarrow \\
\mathsf{loop}(y \otimes x) \\
\| \\
{\scriptstyle \mathsf{loop\text{-}comp}(y,x)^{-1}} \\
\Downarrow \\
\mathsf{loop}(y) \cdot \mathsf{loop}(x) \\
\| \\
{\scriptstyle \mathsf{transp}\ \textit{on constant endpoint}} \\
\Downarrow \\
\mathsf{transp}^{z \mapsto \mathsf{base} = z}(\mathsf{loop}(x), \mathsf{loop}(y))
\end{gathered}$$

Finally, we construct $\psi_{\mathsf{loop\text{-}comp}}$, whose presence is a key difference between our setting and that of [9]. Let $x, y, z : G$. We want to prove that the outer diagram of (`loop-comp-Diag`), shown on the next page, commutes.

$$
\begin{tikzcd}[column sep=small]
\mathsf{loop}(\mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(y), \mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(x), z))) \arrow[r, equal, "\substack{\mathsf{transp}\ \textit{respects} \\ \textit{path composition}}"] \arrow[d, Rightarrow] & \mathsf{loop}(\mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(x) \cdot \mathsf{loop}(y), z)) \arrow[r, equal, "\textit{via}\ \mathsf{loop\text{-}comp}(x,y)"] \arrow[d, phantom, "\boxed{\mathsf{transp\text{-}codes\text{-}coher}}"] & \mathsf{loop}(\mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(x \otimes y), z)) \arrow[d, Rightarrow] \\
\mathsf{loop}(\mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(x), z) \otimes y) \arrow[r, dashed, equal] \arrow[d, Rightarrow, "\psi_{\mathsf{loop}}"'] \arrow[dr, phantom, "\boxed{\mathsf{hnat}(\mathsf{transp\text{-}codes}(x,z))}"] & \mathsf{loop}((z \otimes x) \otimes y) \arrow[r, dashed, equal] \arrow[d, dashed, equal] & \mathsf{loop}(z \otimes (x \otimes y)) \arrow[dddd, Rightarrow, "\psi_{\mathsf{loop}}"] \\
\mathsf{loop}(\mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(x), z)) \cdot \mathsf{loop}(y) \arrow[r, dashed, equal] \arrow[d, Rightarrow] \arrow[dr, phantom, "\boxed{\mathsf{hnat}(\mathsf{transp\text{-}codes}(x,z))}"] & \mathsf{loop}(z \otimes x) \cdot \mathsf{loop}(y) \arrow[d, dashed, equal] \arrow[ddl, dashed, equal] \arrow[r, phantom, "\boxed{\mathsf{loop\text{-}assoc}(z,x,y)}"] & {} \\
\mathsf{transp}^{\mathsf{base}=z}(\mathsf{loop}(y), \mathsf{loop}(\mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(x), z))) \arrow[d, Rightarrow] & (\mathsf{loop}(z) \cdot \mathsf{loop}(x)) \cdot \mathsf{loop}(y) \arrow[dd, dashed, equal] \arrow[ddl, dashed, equal] \arrow[dl, phantom, "\boxed{\mathsf{hnat}(\mathsf{loop\text{-}comp}(z,x))}"] & \\
\mathsf{transp}^{\mathsf{base}=z}(\mathsf{loop}(y), \mathsf{loop}(z \otimes x)) \arrow[d, Rightarrow, "\psi_{\mathsf{loop}}"'] & & \\
\mathsf{transp}^{\mathsf{base}=z}(\mathsf{loop}(y), \mathsf{loop}(z) \cdot \mathsf{loop}(x)) \arrow[d, Rightarrow] \arrow[dr, phantom, "\boxed{\textit{via path induction}}"] & \mathsf{loop}(z) \cdot \mathsf{loop}(x) \cdot \mathsf{loop}(y) \arrow[r, dashed, equal] \arrow[dd, dashed, equal] \arrow[ddr, phantom, "\boxed{\mathsf{hnat}(\mathsf{loop\text{-}comp}(x,y))}"] & \mathsf{loop}(z) \cdot \mathsf{loop}(x \otimes y) \arrow[dd, Rightarrow] \\
\mathsf{transp}^{\mathsf{base}=z}(\mathsf{loop}(y), \mathsf{transp}^{\mathsf{base}=z}(\mathsf{loop}(x), \mathsf{loop}(z))) \arrow[dr, equal, bend right, "\mathsf{transp}\ \textit{respects path composition}"'] & & \\
& \mathsf{transp}^{\mathsf{base}=z}(\mathsf{loop}(x) \cdot \mathsf{loop}(y), \mathsf{loop}(z)) \arrow[r, equal, "\textit{via}\ \mathsf{loop\text{-}comp}(x,y)"'] & \mathsf{transp}^{\mathsf{base}=z}(\mathsf{loop}(x \otimes y), \mathsf{loop}(z))
\end{tikzcd}
$$

∞

(`loop-comp-Diag`)

It remains to construct the homotopy $\mathsf{transp\text{-}codes\text{-}coher}$, at the top of (`loop-comp-Diag`). This homotopy fills

$$\begin{tikzcd}
\mathsf{loop}(\mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(x)\cdot\mathsf{loop}(y),z)) \arrow[r, equal, "\textit{via } \mathsf{loop\text{-}comp}(x,y)"] \arrow[d, equal, "\mathsf{transp}\ \textit{respects path composition}"'] & \mathsf{loop}(\mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(x\otimes y),z)) \arrow[d, equal, "\mathsf{ap}_{\mathsf{loop}}(\mathsf{transp\text{-}codes}(x\otimes y,z))"] \\
\mathsf{loop}(\mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(y),\mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(x),z))) \arrow[d, equal, "\mathsf{ap}_{\mathsf{loop}}(\mathsf{transp\text{-}codes}(y,\mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(x),z)))"'] & \mathsf{loop}(z\otimes(x\otimes y)) \arrow[d, equal, "\textit{associativity of } \otimes"] \\
\mathsf{loop}(\mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(x),z)\otimes y) \arrow[r, equal, "\textit{via } \mathsf{transp\text{-}codes}(x,z)"'] & \mathsf{loop}((z\otimes x)\otimes y)
\end{tikzcd}$$

which is the image under $\mathsf{ap}_{\mathsf{loop}}$ of a diagram D of paths in G. Thus, it suffices to fill D. By homotopy naturality at $\mathsf{transp\text{-}codes}(x,z)$, the bottom left corner of D fits into the commuting square

$$\begin{tikzcd}
\mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(y),\mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(x),z)) \arrow[r, equal, "\textit{via } \mathsf{transp\text{-}codes}(x,z)"] \arrow[d, equal, "\mathsf{transp\text{-}codes}(y,\mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(x),z))"'] & \mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(y),z\otimes x) \arrow[d, equal, "\mathsf{transp\text{-}codes}(y,z\otimes x)"] \\
\mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(x),z)\otimes y \arrow[r, equal, "\textit{via } \mathsf{transp\text{-}codes}(x,z)"'] & (z\otimes x)\otimes y
\end{tikzcd}$$

After we use this square to rewrite D, we rewrite each of the three paths making up $\mathsf{transp\text{-}codes}(x\otimes y,z)$, at the top right of D:

$$\begin{tikzcd}
\mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(x\otimes y),z) \arrow[d, equal, "\textit{via path induction on } \mathsf{loop}(x\otimes y)"] \\
\mathsf{coe}(\mathsf{ap}_{\mathsf{pr}_1}(\mathsf{ap}_{\mathsf{codes}}(\mathsf{loop}(x\otimes y))),z) \arrow[d, equal, "\textit{via } \mathsf{codes}\textit{'s point computation rule}"] \\
\mathsf{coe}(\zeta_{\mathsf{map}}(x\otimes y),z) \arrow[d, equal, "\textit{univalence axiom}"] \\
z\otimes(x\otimes y)
\end{tikzcd}$$

Call these paths p_0, p_1, and p_2, respectively. First, rewrite p_0 with homotopy naturality:

$$\begin{tikzcd}
\mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(x\otimes y),z) \arrow[r, equal] \arrow[d, equal, "p_0"'] \arrow[dr, phantom, "\mathsf{hnat}(\mathsf{loop\text{-}comp}(x,y))"] & \mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(x)\cdot\mathsf{loop}(y),z) \arrow[d, equal] \\
\mathsf{coe}(\mathsf{ap}_{\mathsf{pr}_1}(\mathsf{ap}_{\mathsf{codes}}(\mathsf{loop}(x\otimes y))),z) \arrow[r, equal] & \mathsf{transp}^{\mathsf{codes}_0}(\mathsf{loop}(x)\cdot\mathsf{loop}(y),z)
\end{tikzcd}$$

Second, rewrite p_1 with codes's tensor computation rule, mechanized at [7, codes-β-mu]. This rule gives us the commuting diagram

$$\begin{tikzcd}
\mathsf{ap}_{\mathsf{pr}_1}(\mathsf{ap}_{\mathsf{codes}}(\mathsf{loop}(x \otimes y))) \arrow[rr, equals, "p_1"] \arrow[d, equals, "\textit{via } \mathsf{loop\text{-}comp}(x,y)"'] & & \zeta_{\mathsf{map}}(x \otimes y) \arrow[d, equals, "\textit{associativity of } \otimes"] \\
\mathsf{ap}_{\mathsf{pr}_1}(\mathsf{ap}_{\mathsf{codes}}(\mathsf{loop}(x) \cdot \mathsf{loop}(y))) \arrow[d, equals, "\textit{via path induction on } \mathsf{loop}(x)"'] & & \zeta_{\mathsf{map}}(x) \cdot \zeta_{\mathsf{map}}(y) \arrow[d, equals, "\textit{via } \mathsf{codes}\textit{'s point computation rule at } y"] \\
\mathsf{ap}_{\mathsf{pr}_1}(\mathsf{ap}_{\mathsf{codes}}(\mathsf{loop}(x))) \cdot \mathsf{ap}_{\mathsf{pr}_1}(\mathsf{ap}_{\mathsf{codes}}(\mathsf{loop}(y))) \arrow[rr, equals, "\textit{via } \mathsf{codes}\textit{'s point computation rule at } x"'] & & \zeta_{\mathsf{map}}(x) \cdot \mathsf{ap}_{\mathsf{pr}_1}(\mathsf{ap}_{\mathsf{codes}}(\mathsf{loop}(y)))
\end{tikzcd}$$

Finally, rewrite p_2 with Lemma 3.0.2.

Now, by routine (though messy) path algebra, we can prove that D commutes by cancelling the $\mathsf{loop\text{-}comp}$ terms, codes's point computation terms, the univalence terms, and the terms defined by path induction on loop. This completes the definition of decode. Hence loop is an equivalence [7, loop-equiv].

Remark 3.0.4. Our proof of delooping is an extension of [3, Section 4.3], which shows the result when G is an (ordinary) group. The difference between our proof and that of [3] is that when G is a group,

- the target of the recursion defining codes is a 1-type, namely Set, and
- the construction of $\mathsf{transp\text{-}codes\text{-}coher}$ is trivial, because G is a set.

Our formalization, however, is completely separate from that of [3].

Open problem. Prove that every 2-group G is an infinite loop space by constructing an Ω-spectrum $G, K_2(G), \ldots$. Much, but not all, of the framework of [9] can be adapted to the 2-group setting.

4 The delooping functor as a biequivalence

In this section, we make K_2 into a pseudofunctor. Then we use Section 3 to show that, together with the loop space pseudofunctor, the pseudofunctor K_2 fits into a biequivalence between **2Grp** and $\mathbf{2Type_0^*}$.

Definition 4.0.1 ([7, PsfunctorStr]). Let $\mathcal{C}$ and $\mathcal{D}$ be bicategories. A *pseudofunctor from $\mathcal{C}$ to $\mathcal{D}$* is a function $F_0 : \mathsf{Ob}(\mathcal{C}) \to \mathsf{Ob}(\mathcal{D})$ together with

- a function $F_1 : \mathsf{hom}_{\mathcal{C}}(a,b) \to \mathsf{hom}_{\mathcal{D}}(F_0(a), F_0(b))$ for all $a, b : \mathsf{Ob}$, called the *action on morphisms*
- a 2-cell $F_{\mathsf{id}}(a) : F_1(\mathsf{id}_a) = \mathsf{id}_{F_0(a)}$ for each $a : \mathsf{Ob}$
- a 2-cell $F_\circ(f,g) : F_1(g \circ f) = F_1(g) \circ F_1(f)$ for all composable morphisms f and g
- coherence identities witnessing that $F_\circ$ commutes with the right unitors, with the left unitors, and with the associators.

Example 4.0.2. We equip the object function $K_2 : \mathsf{Ob}(\mathbf{2Grp}) \to \mathsf{Ob}(\mathbf{2Type}_{\mathbf{0}}^{*})$ with the structure of a pseudofunctor. Its action on morphisms [7, K_2-map] is defined by sending $G_1 \xrightarrow{f} G_2$ to the pointed map defined by K_2-recursion on the composite 2-group morphism

$$G_1 \longrightarrow G_2 \longrightarrow \Omega(K_2(G_2))$$

This action preserves the identity morphism [7, KFunctor-idf] as well as composition [7, KFunctor-comp], with both preservation proofs defined by K_2-induction in the form of Lemma 4.0.7, below. We prove the coherence identities with unitors at [7, KFunctor-conv-unit] and with the associator at [7, KFunctor-conv-assoc].

The action on 2-cells can be put in an extensional form $\mathsf{2c\text{-}act}_{K_2}$ taking natural isomorphisms of 2-group morphisms to pointed homotopies [7, apK_2]. The function $\mathsf{2c\text{-}act}_{K_2}$ is defined by Lemma 4.0.7, below.

Example 4.0.3. The loop space Ω forms a pseudofunctor $\mathbf{2Type}_{\mathbf{0}}^{*} \to \mathbf{2Grp}$, whose object function and actions of morphisms are defined as in Example 2.0.9. We prove the coherence identities for Ω at [7, LoopFunctor-conv]. As for K_2, the action on 2-cells can be put in an extensional form $\mathsf{2c\text{-}act}_{\Omega}$ [7, LoopFunctor-ap], which takes pointed homotopies to natural isomorphisms of 2-group morphisms. It is defined by induction on pointed homotopies [7, $\odot$hom-ind], a form of the SIP for pointed maps.

Lemma 4.0.4 ([7, Ω-fmap-ap-hnat]). *Let $f :\equiv (f_0, f_p)\,, g :\equiv (g_0, g_p) : (X, x_0) \to_* Y$ be morphisms in $\mathbf{2Type}_{\mathbf{0}}^{*}$. Let $H :\equiv (H_0, H_p)$ be a pointed homotopy between f and g. The underlying homotopy of $\theta_H :\equiv \mathsf{2c\text{-}act}_{\Omega}(H)$ fits into a commuting pentagon*

$$\begin{tikzcd}
\mathsf{fun}(\Omega(f))(x) \arrow[rr, equal, "\theta_H(x)"] \arrow[d, equal, "\substack{\textit{propositional } \beta\textit{-rule} \\ \textit{for } \Omega(f)}"'] & & \mathsf{fun}(\Omega(g))(x) \arrow[d, equal, "\substack{\textit{propositional } \beta\textit{-rule} \\ \textit{for } \Omega(g)}"] \\
f_p^{-1} \cdot \mathsf{ap}_f(p) \cdot f_p \arrow[dr, equal, "\textit{via } \mathsf{hnat}(p)"'] & & g_p^{-1} \cdot \mathsf{ap}_g(p) \cdot g_p \arrow[dl, equal, "\textit{via } H_p"] \\
& f_p^{-1} \cdot \left(H_0(x_0) \cdot \mathsf{ap}_g(p) \cdot H_0(x_0)^{-1}\right) \cdot f_p &
\end{tikzcd}$$

for each loop $p : \Omega(X)$.

For every bicategory $\mathcal{C}$, we can form the identity pseudofunctor $\mathsf{id}_{\mathcal{C}} : \mathcal{C} \to \mathcal{C}$ [7, idfBCσ]. Its object function is the identity, as are its actions on 1-cells and 2-cells. We also can form the composite $G \circ F$ of pseudofunctors [7, $\circ$BCσ]. Its object function is the composite $G_0 \circ F_0$. Its action on morphisms is $G_1 \circ F_1$. Its action on 2-cells is $\mathsf{ap}_{G_1} \circ \mathsf{ap}_{F_1}$.

Definition 4.0.5 ([7, Biequiv]). Let $\mathcal{C}$ and $\mathcal{D}$ be bicategories.

(1) Let $F : \mathcal{C} \to \mathcal{D}$ and $G : \mathcal{D} \to \mathcal{C}$ be pseudofunctors. A *pseudotransformation from F to G* consists of

- a *component* 1-cell $\eta_0(a) : F_0(a) \to G_0(a)$ for each $a : \mathsf{Ob}(\mathcal{C})$
- a 2-cell $\eta_1(f)$ making the square

$$\begin{array}{ccc} F_0(a) & \xrightarrow{F_1(f)} & F_0(b) \\ {\scriptstyle \eta_0(a)}\big\downarrow & & \big\downarrow{\scriptstyle \eta_0(b)} \\ G_0(a) & \xrightarrow[G_1(f)]{} & G_0(b) \end{array}$$

  commute for all $a, b : \mathsf{Ob}(\mathcal{C})$ and $f : \mathsf{hom}_{\mathcal{C}}(a, b)$.
- a coherence identity witnessing that η_1 commutes with the unitors and one witnessing that it commutes with the associators.

The type of such pseudotransformations is denoted by $F \Rightarrow G$.

(2) A *biequivalence between* $\mathcal{C}$ *and* $\mathcal{D}$ is a pseudofunctor $F : \mathcal{C} \to \mathcal{D}$ together with

- a pseudofunctor $G : \mathcal{D} \to \mathcal{C}$
- a pseudotransformation $\tau_1 : F \circ G \Rightarrow \mathsf{id}_{\mathcal{D}}$ each of whose components is an adjoint equivalence in $\mathcal{D}$
- a pseudotransformation $\tau_2 : \mathsf{id}_{\mathcal{C}} \Rightarrow G \circ F$ each of whose components is an adjoint equivalence in $\mathcal{C}$.

Note 4.0.6.

- Our definition of *pseudotransformation* does not explicitly require η_1 to commute with 2-cells. For us, however, this property follows from homotopy naturality as 2-cells are paths.
- Our definition of *biequivalence* is a direct generalization of the 1-categorical notion. By [8, Proposition 6.2.16], it is logically equivalent to the definition in terms of *modifications* [1, Definition 2.17].

The next two lemmas follow from K_2's induction principle. The first gives us a way to build a homotopy between two functions defined by K_2-recursion. The second gives us a way to prove that two such homotopies are themselves pointwise equal. The first lemma is useful for constructing 2-cells in $\mathbf{2Type}_0^*$ required by Definition 4.0.5(1). The second lemma is useful for proving the coherence identities also required by Definition 4.0.5(1).

Lemma 4.0.7 ([7, K-hom-ind]). *Let G be a 2-group and X be a 2-type. Let $f, g : K_2(G) \to X$. Given*

$$\begin{aligned} \mathsf{base}^{\sim} &: f(\mathsf{base}) = g(\mathsf{base}) \\ \mathsf{loop}^{\sim} &: \prod_{x:G} \mathsf{ap}_f(\mathsf{loop}(x)) \cdot \mathsf{base}^{\sim} = \mathsf{base}^{\sim} \cdot \mathsf{ap}_g(\mathsf{loop}(x)) \\ \mathsf{loop\text{-}comp}^{\sim} &: \mathsf{loop}^{\sim} \textit{ commutes with } G\textit{'s tensor product} \end{aligned}$$

we have a homotopy $H : f \sim g$ satisfying $H(\mathsf{base}) \equiv \mathsf{base}^{\sim}$ and the propositional β-rule

$$\begin{tikzcd} \bullet \arrow[r, equal] \arrow[d, equal] \arrow[dr, phantom, "\mathsf{hnat}(\mathsf{loop}(x))"] & \bullet \arrow[d, equal] \\ \bullet \arrow[r, equal] & \bullet \end{tikzcd} \quad = \quad \begin{tikzcd} \bullet \arrow[r, equal] \arrow[d, equal] \arrow[dr, phantom, "\mathsf{loop}^{\sim}(x)"] & \bullet \arrow[d, equal] \\ \bullet \arrow[r, equal] & \bullet \end{tikzcd}$$

between commuting squares for each $x : G$.

Lemma 4.0.8 ([7, K-hom2-ind]). *Let G, X, f, and g be as in Lemma 4.0.7. Let $H_1, H_2 : f \sim g$. Suppose that we have an identity $\mathsf{base}^{\sim\sim} : H_1(\mathsf{base}) = H_2(\mathsf{base})$ and a 3-dimensional path $\mathsf{loop}^{\sim\sim}$*

$$\begin{tikzcd} \bullet \arrow[r, equal, "H_1(\mathsf{base})"] \arrow[d, equal] \arrow[dr, phantom, "\mathsf{hnat}(\mathsf{loop}(x))"] & \bullet \arrow[d, equal] \\ \bullet \arrow[r, equal, "H_1(\mathsf{base})"'] & \bullet \end{tikzcd} \quad = \quad \begin{tikzcd} \bullet \arrow[r, equal, "H_2(\mathsf{base})"] \arrow[d, equal] \arrow[dr, phantom, "\mathsf{hnat}(\mathsf{loop}(x))"] & \bullet \arrow[d, equal] \\ \bullet \arrow[r, equal, "H_2(\mathsf{base})"'] & \bullet \end{tikzcd} \quad \cdot \quad \begin{tikzcd} \bullet \arrow[d, Rightarrow] & \bullet \arrow[l, Rightarrow, "H_2(\mathsf{base})"'] \arrow[r, Rightarrow, "H_1(\mathsf{base})"] \arrow[d, phantom, "\mathit{via}\ \mathsf{base}^{\sim\sim}"] & \bullet \arrow[d, Rightarrow] \\ \bullet \arrow[r, Rightarrow, "H_2(\mathsf{base})^{-1}"'] & \bullet & \bullet \arrow[l, Rightarrow, "H_1(\mathsf{base})^{-1}"] \end{tikzcd}$$

Then we have a homotopy $L : H_1 \sim H_2$ satisfying $L(\mathsf{base}) \equiv \mathsf{base}^{\sim\sim}$.

The next result lets us build one of the families of adjoint equivalences required by Definition 4.0.5(2). (Section 3 provides the other such family.)

Note 4.0.9. Let X be a pointed connected 2-type. Define the pointed map $\varphi_X : K_2(\Omega(X)) \to_* X$ by K_2-recursion on the identity 2-group morphism $\Omega(X) \to \Omega(X)$. By φ_X's point computation rule, the triangle of types

$$\begin{tikzcd} & \Omega(X) \arrow[dl, "\mathsf{loop}"'] \arrow[dr, "\mathsf{id}"] & \\ \Omega(K_2(\Omega(X))) \arrow[rr, "\mathsf{fun}(\Omega(\varphi_X))"'] & & \Omega(X) \end{tikzcd}$$

commutes [7, LoopK-hom]. By Section 3, loop is an equivalence, so that $\mathsf{fun}(\Omega(\varphi_X))$ is one as well. Since both X and $K_2(\Omega(X))$ are connected, it follows that φ_X is an equivalence [7, Loop-conn-equiv].

Theorem 4.0.10 ([7, Biequiv-main]). *The pseudofunctors K_2 and Ω form a biequivalence between* $\mathbf{2Grp}$ *and* $\mathbf{2Type}_{\mathbf{0}}^{*}$.

Proof sketch.

Step 1: *Construct $\tau_1 : K_2 \circ \Omega \Rightarrow \mathsf{id}_{\mathbf{2Type}_{\mathbf{0}}^{*}}$.*

For each pointed connected 2-type X, define the pointed map $\eta_0^1(X) : K_2(\Omega(X)) \to_* X$ by K_2-recursion on the identity 2-group morphism $\Omega(X) \to \Omega(X)$. Let $f : X \to_* Y$ be a morphism in

$\mathbf{2Type}_{\mathbf{0}}^{*}$. We want a path $\eta_1^1(f)$ making the square of pointed maps

$$\begin{tikzcd}
K_2(\Omega(X)) \arrow[r, "K_2(\Omega(f))"] \arrow[d, "\eta_0^1(X)"'] & K_2(\Omega(Y)) \arrow[d, "\eta_0^1(Y)"] \\
X \arrow[r, dash, "f"'] & Y
\end{tikzcd}$$

commute. By the SIP for pointed maps, it suffices to find a homotopy

$$H_1(f) \quad : \quad \mathsf{fun}(f) \circ \mathsf{fun}(\eta_0^1(X)) \;\sim\; \mathsf{fun}(\eta_0^1(Y)) \circ \mathsf{fun}(K_2(\Omega(f)))$$

between the underlying functions along with a dependent path $H_2(f)$ over $H_1(f)$ between the corresponding proofs of pointedness. We define $H_1(f)$ by applying Lemma 4.0.7 to the data

$$\begin{aligned}
\mathsf{base}^{\sim} &\;\coloneqq\; \mathsf{refl} \\
\mathsf{loop}^{\sim} &\;\coloneqq\; H_1(f)\text{-}\mathsf{loop} \\
\mathsf{loop\text{-}comp}^{\sim} &\;\coloneqq\; \textit{defined at [7, SqKLoop-coher]}
\end{aligned}$$

Here, $H_1(f)$-$\mathsf{loop}(p)$ is defined as the chain of paths

$$\begin{tikzcd}
\mathsf{ap}_{\mathsf{fun}(f \circ \eta_0^1(X))}(\mathsf{loop}(p)) \arrow[d, Rightarrow, "\textit{via } \eta_0^1(X)\textit{'s point computation rule}"'] \\
\mathsf{ap}_{\mathsf{fun}(f)}(p) \arrow[d, Rightarrow, "\textit{via } \eta_0^1(Y)\textit{'s point computation rule}"'] \\
\mathsf{ap}_{\mathsf{fun}(\eta_0^1(Y))}(\mathsf{loop}(\mathsf{ap}_{\mathsf{fun}(f)}(p))) \arrow[d, Rightarrow, "\textit{via } K_2(\Omega(f))\textit{'s point computation rule}"'] \\
\mathsf{ap}_{\mathsf{fun}(\eta_0^1(Y) \circ K_2(\Omega(f)))}(\mathsf{loop}(p))
\end{tikzcd}$$

for each $p : x_0 = x_0$ where x_0 denotes the basepoint of X. Our definition of $H_1(f)$ makes it trivial to define $H_2(f)$. This completes the construction of $\eta_1^1(f)$ [7, SqKLoop].

We must verify that η_1^1 satisfies the relevant coherence identities. In the case of unitors, we must prove that

$$\begin{tikzcd}
\mathsf{id}_X \circ \eta_0^1(X) \arrow[r, Rightarrow, "\eta_1^1(\mathsf{id}_X)"] \arrow[d, Rightarrow, "\textit{left unitor}"'] & \eta_0^1(X) \circ K_2(\Omega(\mathsf{id}_X)) \arrow[d, Rightarrow, "\textit{composite of } K_2\textit{'s } \mathsf{id} \textit{ preservation with } \Omega\textit{'s } \mathsf{id} \textit{ preservation}"] \\
\eta_0^1(X) \arrow[r, Rightarrow, "\textit{right unitor}"'] & \eta_0^1(X) \circ \mathsf{id}_{K_2(\Omega(X))}
\end{tikzcd} \qquad (\texttt{unitor-coher1})$$

commutes for each pointed connected 2-type X. By the SIP for pointed homotopies [7, ⊙→∼-ind], this amounts to a homotopy $M_1(X)$ between the homotopies underlying the 2-cells in (`unitor-coher1`) along with a dependent path $M_2(X)$ over $M_1(X)$ between the corresponding proofs of pointedness.[1] We define $M_1(X)$ by applying Lemma 4.0.8 to the data

$$\begin{aligned} \mathsf{base}^{\sim\sim} &:= \mathsf{refl} \\ \mathsf{loop}^{\sim\sim} &:= M_1(X)\text{-}\mathsf{loop} \end{aligned}$$

Here, for each loop $p : x_0 = x_0$, $M_1(X)$-$\mathsf{loop}(p)$ is an identity

$$\begin{tikzcd}
\mathsf{base} \arrow[r, equal, "H_1(\mathsf{id}_X, \mathsf{base})"] \arrow[d, equal, "\mathsf{ap}_{\mathsf{fun}(\eta_0^1(X))}(\mathsf{loop}(p))"'] & \mathsf{base} \arrow[r, equal, "via\ \mathsf{2c\text{-}act}_{K_2}(\mathsf{id}_{\mathsf{id}_{\Omega(X)}}, \mathsf{base})"] \arrow[d, equal, dashed] \arrow[dr, phantom, "NatSq_1"] & \mathsf{base} \arrow[r, equal, "\mathsf{id}\ preservation\ of\ K_2"] \arrow[d, equal, dashed] & \mathsf{base} \arrow[d, equal, "\mathsf{ap}_{\mathsf{fun}(\eta_0^1(X))}(\mathsf{loop}(p))"] \\
\mathsf{base} \arrow[r, equal, "H_1(\mathsf{id}_X, \mathsf{base})"'] & \mathsf{base} \arrow[r, equal, "via\ \mathsf{2c\text{-}act}_{K_2}(\mathsf{id}_{\mathsf{id}_{\Omega(X)}}, \mathsf{base})"'] & \mathsf{base} \arrow[r, equal, "\mathsf{id}\ preservation\ of\ K_2"'] & \mathsf{base}
\end{tikzcd}$$

$$\Vert$$

$$\begin{tikzcd}
\mathsf{base} \arrow[r, equal, "\mathsf{refl}"] \arrow[d, equal, "\mathsf{ap}_{\mathsf{fun}(\eta_0^1(X))}(\mathsf{loop}(p))"'] \arrow[dr, phantom, "NatSq_2"] & \mathsf{base} \arrow[d, equal, "\mathsf{ap}_{\mathsf{fun}(\eta_0^1(X))}(\mathsf{loop}(p))"] \\
\mathsf{base} \arrow[r, equal, "\mathsf{refl}"'] & \mathsf{base}
\end{tikzcd}$$

between homotopy-naturality squares at $\mathsf{loop}(p)$, where $\mathsf{2c\text{-}act}_{K_2}$ is as in Example 4.0.2. Note that the upper three (hence bottom three) paths of the upper square reduce to refl by the base computation rule of K_2-induction. To construct $M_1(X)$-$\mathsf{loop}(p)$, we decompose $NatSq_1$ into three loops

$$L_1(p), L_2(p), L_3(p) \ : \ \mathsf{ap}_{\mathsf{fun}(\eta_0^1(X))}(\mathsf{loop}(p)) = \mathsf{ap}_{\mathsf{fun}(\eta_0^1(X))}(\mathsf{loop}(p))$$

corresponding to the three homotopy naturality sub-squares, from left to right [7, KLoop-ptr-idf-aux1]. This decomposition is possible because homotopy naturality preserves path composiiton. By

[1] Although (`unitor-coher1`) is a proposition, the target of $M_1(X)$ is just a set. Hence the induction principle for connected types does not apply here.

the β-rule of Lemma 4.0.7, we have commuting diagrams

$$
\begin{tikzcd}
\mathsf{ap}_{\mathsf{fun}(\mathsf{id}_X \circ \eta_0^1(X))}(\mathsf{loop}(p)) \arrow[r, equal, bend left, "L_1(p)"] \arrow[r, Rightarrow, bend right, "H_1(f\text{-}\mathsf{loop}(p))"'] & \mathsf{ap}_{\mathsf{fun}(\eta_0^1(X) \circ K_2(\Omega(\mathsf{id}_X)))}(\mathsf{loop}(p))
\end{tikzcd}
$$

$$
\begin{tikzcd}
\mathsf{ap}_{\mathsf{fun}(\eta_0^1(X))}(\mathsf{loop}(\mathsf{ap}_{\mathsf{id}_{\Omega(X)}}(p))) \arrow[r, equal, "\mathsf{ap}_{\mathsf{id}} \textit{ is identity}"] \arrow[d, equal, "\textit{via } K_2(\Omega(\mathsf{id}_X))\textit{'s point computation rule}"'] & \mathsf{ap}_{\mathsf{fun}(\eta_0^1(X))}(\mathsf{loop}(p)) \arrow[d, equal, "\textit{via } K_2(\mathsf{id}_{\Omega(X)})\textit{'s point computation rule}"] \\
\mathsf{ap}_{\mathsf{fun}(\eta_0^1(X))}(\mathsf{ap}_{\mathsf{fun}(K_2(\Omega(\mathsf{id}_X)))}(\mathsf{loop}(p))) \arrow[r, equal, "L_2(p)"'] & \mathsf{ap}_{\mathsf{fun}(\eta_0^1(X))}(\mathsf{ap}_{\mathsf{fun}(K_2(\mathsf{id}_{\Omega(X)}))}(\mathsf{loop}(p)))
\end{tikzcd}
$$

$$
\begin{tikzcd}
\mathsf{ap}_{\mathsf{fun}(\eta_0^1(X))}(\mathsf{ap}_{\mathsf{fun}(K_2(\mathsf{id}_{\Omega(X)}))}(\mathsf{loop}(p))) \arrow[r, equal, "\textit{via } K_2(\mathsf{id}_{\Omega(X)})\textit{'s point computation rule}"] \arrow[dr, Rightarrow, "L_3(p)"'] & \mathsf{ap}_{\mathsf{fun}(\eta_0^1(X))}(\mathsf{loop}(p)) \arrow[d, equal, "\mathsf{ap}_{\mathsf{id}} \textit{ is identity}"] \\
& \mathsf{ap}_{\mathsf{fun}(\eta_0^1(X))}(\mathsf{ap}_{\mathsf{id}_{K_2(\Omega(X))}}(\mathsf{loop}(p)))
\end{tikzcd}
$$

which are mechanized at [7, KLoop-ptr-idf-aux0]. We further adjust $L_1(p)$ by rewriting its middle path via homotopy naturality:

$$
\begin{tikzcd}
\mathsf{ap}_{\mathsf{id}_{\Omega(X)}}(p) \arrow[r, equal, "\mathsf{ap}_{\mathsf{id}} \textit{ is identity}"] \arrow[d, equal, "\boldsymbol{\eta_0^1(X)\textit{'s point computation rule at }} \mathsf{ap}_{\mathsf{id}_{\Omega(X)}}(p)"'] & p \arrow[d, equal, "\eta_0^1(X)\textit{'s point computation rule at } p"] \\
\mathsf{ap}_{\mathsf{fun}(\eta_0^1(X))}(\mathsf{loop}(\mathsf{ap}_{\mathsf{id}_{\Omega(X)}}(p))) \arrow[r, equal, "\mathsf{ap}_{\mathsf{id}} \textit{ is identity}"'] & \mathsf{ap}_{\mathsf{fun}(\eta_0^1(X))}(\mathsf{loop}(p))
\end{tikzcd}
$$

Returning to $M_1(X)$-$\mathsf{loop}(p)$, notice that *NatSq2* is trivial. Thus, we can derive $M_1(X)$-$\mathsf{loop}(p)$ by proving that the composition of $L_1(p)$, $L_2(p)$, and $L_3(p)$ is trivial. This proof is a routine computation that works by repeatedly cancelling point computation rules [7, KLoop-ptr-idf-coher]. Further, our definition of $M_1(X)$ makes it trivial to define $M_2(X)$. This completes the coherence identity with the unitors [7, KLoop-coher-unit]. The coherence identity with the associator, omitted here, is similar but more complicated [7, KLoop-PT-assoc].

Step 2: *Construct* $\tau_2 : \mathsf{id}_{\mathbf{2Grp}} \Rightarrow \Omega \circ K_2$.

For each 2-group G, define the 2-group morphism $\eta_0^2(G) :\equiv \mathsf{loop} : G \to \Omega(K_2(G))$. Let $f : G_1 \to G_2$ be a morphism of 2-groups. To define $\eta_1^2(f)$, we want a natural isomorphism

$$
\begin{tikzcd}
G_1 \arrow[r, "f"] \arrow[d, "\mathsf{loop}"'] \arrow[dr, phantom, "I(f)"] & G_2 \arrow[d, "\mathsf{loop}"] \\
\Omega(K_2(G_1)) \arrow[r, "\Omega(K_2(f))"'] & \Omega(K_2(G_2))
\end{tikzcd}
$$

of 2-group morphisms. We simply define the underlying homotopy of $I(f)$ as $K_2(f)$'s point computation rule. The fact that this respects the tensor product follows from $K_2(f)$'s tensor computation rule [7, SqLoopK].

We must verify that η_1^2 satisfies the relevant coherence identities. In the case of unitors, we must prove that

$$\begin{tikzcd}
\Omega(K_2(\mathsf{id}_G)) \circ \eta_0^2(G) \arrow[r, equal, "\eta_1^2(\mathsf{id}_G)"] \arrow[d, equal, "\substack{\textit{composite of } \Omega'\textit{s's } \mathsf{id} \\ \textit{preservation with} \\ K_2\textit{'s } \mathsf{id} \textit{ preservation}}"'] & \eta_0^2(G) \circ \mathsf{id}_G \arrow[d, equal, "\textit{right unitor}"] \\
\mathsf{id}_{\Omega(K_2(G))} \circ \eta_0^2(G) \arrow[r, equal, "\textit{left unitor}"'] & \eta_0^2(G)
\end{tikzcd} \tag{unitor-coher2}$$

commutes for each 2-group G. By the SIP for natural isomorphisms [7, natiso∼-ind], the identity (`unitor-coher2`) amounts to a homotopy H_G between the underlying homotopies of the associated natural isomorphisms. For each $x : G$, we define $H_G(x)$ as the commuting outer diagram

$$\begin{tikzcd}
& \mathsf{loop}(x) \arrow[dl, equal, "\mathsf{ap}_{\mathsf{id}} \textit{ is identity}"'] \arrow[dr, equal, "\substack{K_2(\mathsf{id}_G)\textit{'s point} \\ \textit{computation rule}}"] \arrow[d, phantom, "\beta\textbf{\textit{-rule of Lemma 4.0.7}}"] & \\
\mathsf{ap}_{\mathsf{id}_{\Omega(K_2(G))x}}(\mathsf{loop}(x)) \arrow[rr, equal, "\substack{\mathsf{hnat}(\mathsf{loop}(x)) \textit{ at } K_2\textit{'s} \\ \mathsf{id} \textit{ preservation proof}}"'] \arrow[d, equal, "\mathsf{refl}"'] & {} & \mathsf{ap}_{\mathsf{fun}(K_2(\mathsf{id}_G))}(\mathsf{loop}(x)) \arrow[d, equal, "\mathsf{refl}"] \\
\Omega(\mathsf{id}_{K_2(G)})(\mathsf{loop}(x)) \arrow[rr, equal, "\mathsf{2c\text{-}act}_{\Omega}(K_2(\mathsf{id}_G),\mathsf{loop}(x))"'] \arrow[rr, phantom, bend left=30, "\textbf{\textit{Lemma 4.0.4}}"] & & \Omega(K_2(\mathsf{id}_G))(\mathsf{loop}(x))
\end{tikzcd}$$

where $\mathsf{2c\text{-}act}_\Omega$ is as in Example 4.0.3. This completes the coherence identity with the unitors [7, LoopK-PT-unit]. Again, the coherence identity with the associator, omitted here, is similar but more complicated [7, LoopK-PT-assoc].

Step 3: *Prove that both τ_1 and τ_2 are levelwise adjoint equivalences.*

By Note 4.0.9, τ_1 is a levelwise adjoint equivalence. By Section 3, τ_2 is a levelwise adjoint equivalence. This completes the desired biequivalence.

□

5 The delooping functor as an isomorphism

In this section, we show that the pseudofunctor Ω is an *isomorphism* of bicategories, i.e., it is an equivalence on objects and is fully faithful. We do so by proving that every biequivalence between *quasi-univalent* bicategories is an isomorphism. Here the key step is showing that every equivalence of wild categories is fully faithful. We then use univalence to prove that isomorphism is equivalent to identity of bicategories. From this we immediately get an identity between **2Grp** and $\mathbf{2Type}_{\mathbf{0}}^{*}$.

Definition 5.0.1. Let $\mathcal{C}$ and $\mathcal{D}$ be bicategories and let $F : \mathcal{C} \to \mathcal{D}$ be a pseudofunctor. We say that F is an *isomorphism* if $F_0 : \mathsf{Ob}(\mathcal{C}) \to \mathsf{Ob}(\mathcal{D})$ is an equivalence and $F_1 : \mathsf{hom}_{\mathcal{C}}(a, b) \to \mathsf{hom}_{\mathcal{D}}(F_0(a), F_0(b))$ is an equivalence for all $a, b : \mathsf{Ob}(\mathcal{C})$. We denote the type of isomorphisms between $\mathcal{C}$ and $\mathcal{D}$ by $\mathcal{C} \cong \mathcal{D}$.

Lemma 5.0.2 ([7, Bicat-iso]). *For all bicategories $\mathcal{C}$ and $\mathcal{D}$, $(\mathcal{C} \cong \mathcal{D}) \simeq (\mathcal{C} = \mathcal{D})$.*

Proof. By the univalence axiom combined with the SIP. □

Definition 5.0.3. A bicategory $\mathcal{C}$ is *quasi-univalent* if for all $a, b : \mathsf{Ob}(\mathcal{C})$, $\mathsf{AdjEquiv}(a, b) \to (a = b)$.

By Lemma 2.0.12, both **2Grp** and $\mathbf{2Type}_{\mathbf{0}}^{*}$ are quasi-univalent [7, qu-2G *and* qu-Pt02]. It is easy to see that every biequivalence between quasi-univalent bicategories is an equivalence on objects. To see that it is an isomorphism, we must show that it is fully faithful, i.e., its action on 1-cells is a family of equivalences. We show a more general result by moving to wild category theory.[2]

Definition 5.0.4. A morphism $f : A \to B$ in a wild category is an *equivalence* if it has a morphism $g : B \to A$ and identities $\mathsf{id}_A = g \circ f$ and $\mathsf{id}_B = f \circ g$.

Definition 5.0.5 ([7, WildNatTr]). Let $\mathcal{C}$ and $\mathcal{D}$ be wild categories.

- Let $F, G : \mathcal{C} \to \mathcal{D}$ be functors of wild categories and $\tau : F \Rightarrow G$ be a natural transformation. We say that τ is a *natural isomorphism* if its component $\tau_0(X)$ is an equivalence for each $X : \mathsf{Ob}(\mathcal{C})$.

- Let $F : \mathcal{C} \to \mathcal{D}$ be a functor of wild categories. We say that F is an *equivalence* if it has a functor $G : \mathcal{D} \to \mathcal{C}$ along with natural isomorphisms $\epsilon : F \circ G \Rightarrow \mathsf{id}_{\mathcal{D}}$ and $\eta : \mathsf{id}_{\mathcal{C}} \Rightarrow G \circ F$. In this case, we say that F is a *(compontent-wise) half-adjoint equivalence* if it comes with a triangle identity
$$\epsilon_0(F_0(X)) \circ F_1(\eta_0(X)) \;=\; \mathsf{id}_{F_0(X)}$$
for all $X : \mathsf{Ob}(\mathcal{C})$.

Lemma 5.0.6.

(1) Every equivalence of wild categories can be promoted to a half-adjoint equivalence.

(2) Every half-adjoint equivalence of wild categories is fully faithful.

Proof. By arguments similar to the ones for 1-categories. □

Corollary 5.0.7 ([7, Equiv-wc-ff]). *Every equivalence of wild categories is fully faithful.*

[2]See [5, Section 2.2] for the notions of *wild category* and *functor of wild categories.*

Lemma 5.0.8 ([7, biequiv-to-iso]). *Every biequivalence between quasi-univalent bicategories is an isomorphism.*

Proof. Let F be such a biequivalence. We have already observed that it's an equivalence on objects. Now, every bicategory has an underlying wild category, and clearly every biequivalence induces an equivalence of the underlying wild categories with the same action on morphisms. Thus, by applying Corollary 5.0.7 to the equivalence of wild categories induced by F, we find that F is fully faithful. $\square$

Theorem 5.0.9 ([7, 2Grp-Ptd02-eql]). *The pseudofunctor* Ω *induces an identity* $\mathbf{2Grp} = \mathbf{2Type}_{\mathbf{0}}^{*}$ *of bicategories.*

Proof. By Lemma 5.0.2 combined with Lemma 5.0.8 and Theorem 4.0.10. $\square$

Remark 5.0.10. To prove Theorem 5.0.9, we only need that Ω is a pseudofunctor and that it forms a 1-*coherent* equivalence with K_2. Notably, we do not use

- the coherence identities of τ_1 or of τ_2 required by Definition 4.0.5
- the coherence identities of K_2 required by Definition 4.0.1.

This means that we can avoid the most difficult computations to construct the induced identity of bicategories! Of course, we also can use path induction on this identity to show that Ω is part of a biequivalence bieq-from-id.

Still, the explicit biequivalence provided by Theorem 4.0.10 is valuable. Indeed, besides the inverse of its object function, the form that bieq-from-id takes is quite hard to recover in general.[3] Even if fully recovered, it may be much less practical and desirable to work with in the present case than the data provided by our explicit biequivalence.

[3]See [8, Theorem 7.4.1] for a way of building a biequivalence from a weak equivalence (hence from an isomorphism) of bicategories. A type-theoretic version of this theorem is conjectured for univalent bicategories [1, Conjecture 5.7], but it is not yet proven, to our knowledge.

A A short definition of a 2-group morphism

Let G_1 and G_2 be 2-groups and $f_0 : G_1 \to G_2$ be a function between their underlying types. We prove that the function ($\mathtt{ForgMap}(f_0)$) is an equivalence of types. This amounts to showing that if f_0 has the data D_s making up the short definition of 2-*group morphism*, then it also has both unique preservation data for $(-)^{-1}$, which we call P_i, and unique preservation data for id, which we call P_u.

Suppose that f_0 has the data D_s. The data P_u consists of a path $u : \mathsf{id} = f_0(\mathsf{id})$ such that the diagrams

$$\begin{tikzcd}
f_0(x) \arrow[r, equal, "\rho(f_0(x))"] \arrow[d, equal, "\mathsf{ap}_{f_0}(\rho(x))"'] \arrow[dr, phantom, "r_u(x)"] & f_0(x) \otimes \mathsf{id} \arrow[d, equal, "\mathsf{ap}_{f_0(x)\otimes -}(u)"] \\
f_0(x \otimes \mathsf{id}) \arrow[r, equal, "\mu_{x,\mathsf{id}}"'] & f_0(x) \otimes f_0(\mathsf{id})
\end{tikzcd}
\qquad
\begin{tikzcd}
f_0(x) \arrow[r, equal, "\lambda(f_0(x))"] \arrow[d, equal, "\mathsf{ap}_{f_0}(\lambda(x))"'] \arrow[dr, phantom, "\ell_u(x)"] & \mathsf{id} \otimes f_0(x) \arrow[d, equal, "\mathsf{ap}_{-\otimes f_0(x)}(u)"] \\
f_0(\mathsf{id} \otimes x) \arrow[r, equal, "\mu_{\mathsf{id},x}"'] & f_0(\mathsf{id}) \otimes f_0(x)
\end{tikzcd}$$

commute for each $x : G_1$. The data P_i consists of a path $i_x : f_0(x)^{-1} = f_0(x^{-1})$ for each $x : G_1$ such that the diagrams

$$\begin{tikzcd}
f_0(x) \otimes f_0(x)^{-1} \arrow[r, equal, "\mathsf{ap}_{f_0(x)\otimes -}(i_x)"] \arrow[d, equal, "\mathsf{rinv}(f_0(x))"'] & f_0(x) \otimes f_0(x^{-1}) \arrow[r, equal, "\mu_{x,x^{-1}}"] \arrow[d, phantom, "r_i(x)"] & f_0(x \otimes x^{-1}) \arrow[d, equal, "\mathsf{ap}_{f_0}(\mathsf{rinv}(x))"] \\
\mathsf{id} \arrow[rr, equal, "u"'] & & f_0(\mathsf{id})
\end{tikzcd}$$

$$\begin{tikzcd}
f_0(x)^{-1} \otimes f_0(x) \arrow[r, equal, "\mathsf{ap}_{-\otimes f_0(x)}(i_x)"] \arrow[d, equal, "\mathsf{linv}(f_0(x))"'] & f_0(x^{-1}) \otimes f_0(x) \arrow[r, equal, "\mu_{x^{-1},x}"] \arrow[d, phantom, "\ell_i(x)"] & f_0(x^{-1} \otimes x) \arrow[d, equal, "\mathsf{ap}_{f_0}(\mathsf{linv}(x))"] \\
\mathsf{id} \arrow[rr, equal, "u"'] & & f_0(\mathsf{id})
\end{tikzcd}$$

commute for each $x : G_1$. Note that both $f_0(x) \otimes - : G_2 \to G_2$ and $- \otimes f_0(x) : G_2 \to G_2$ are equivalences of types [7, mu-pre-iso *and* mu-post-iso]. Thus, we have a unique choice of u satisfying $r_u(\mathsf{id})$. Moreover, we have a choice i_{left} of i satisfying ℓ_i and a choice i_{right} of i satisfying r_i. (These two choices happen to be unique.)

Let $x : G_1$. As both ℓ_u and r_u are families of propositions, to recover P_u, we just need to show that our choice of u satisfies $\ell_u(x)$ and $r_u(x)$. First, we show that $r_u(\mathsf{id})$ implies $\ell_u(x)$ [7, rho-to-lam]. Second, we show that $\ell_u(x)$ implies $r_u(x)$ [7, lam-to-rho], so that $r_u(\mathsf{id})$ implies both $r_u(x)$ [7, rhoid-to-rho] and $\ell_u(x)$. The formal proofs of both steps provide the details with explicit equational reasoning, which largely matches the style of a pen-and-paper proof thanks to Agda'a instance search.

It remains to recover P_i. This process works by making a unique choice of i satsifying r_i and then showing that this choice also satisfies ℓ_i. (We could switch the roles of r_i and ℓ_i.) The process relies on the data P_u, which we have already recovered. We refer the reader to either our mechanized proof [7, rinv-to-linv] or [2, Theorem 6.1] for the details.[4]

Remark A.0.1. By keeping track of indices of hom-objects, it's easy to extend our proof to pseudofunctors of *bigroupoids* [10]. (A 2-group is precisely a single-object bigroupoid.) This means that a pseudofunctor $F : B \to C$ of bigroupoids has a simple definition:

- a function $F_0 : B_0 \to C_0$
- an action $F_1 : B_1(a, b) \to C_1(F_0(a), F_0(b))$ on 1-cells
- an action on 2-cells respecting vertical composition
- a 2-cell $F_c(f, g) : F_1(g \circ f) \Rightarrow F_1(g) \circ F_1(f)$ for all composable 1-cells f and g such that F_c respects the associator.

In other words, F is simply a 2-*semifunctor*.[5]

[4] The only reference we have found mentioning that it's possible to recover P_i and P_u simultaneously is [6, Section 2.3]. The author, however, leaves the proof to the reader.

[5] See [3, Chapter 5] for a wonderful look at 2-semifunctors inside HoTT.

References

[1] Benedikt Ahrens, Dan Frumin, Marco Maggesi, Niccolò Veltri, and Niels van der Weide. Bicategories in univalent foundations. *Mathematical Structures in Computer Science*, 31(10):1232–1269, 2021. doi:10.1017/S0960129522000032.

[2] John C. Baez and Aaron D. Lauda. Higher-dimensional algebra. V: 2-Groups. *Theory and Applications of Categories*, 12:423–491, 2004. URL: http://eudml.org/doc/124217.

[3] Tim Baumann. The cup product on cohomology groups in Homotopy Type Theory. Master's thesis, University of Augsburg, 2018. URL: https://timjb.gitlab.io/masters-thesis/cp-in-hott.pdf.

[4] Ulrik Buchholtz, Floris van Doorn, and Egbert Rijke. Higher Groups in Homotopy Type Theory. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS '18, page 205–214, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3209108.3209150.

[5] J Daniel Christensen and Luis Scoccola. The Hurewicz theorem in homotopy type theory. *Algebraic and Geometric Topology*, 23(5):2107–2140, 2023. doi:https://doi.org/10.2140/agt.2023.23.2107.

[6] Josep Elgueta. Permutation 2-groups I: Structure and splitness. *Advances in Mathematics*, 258:286–350, 2014. URL: https://www.sciencedirect.com/science/article/pii/S0001870814001054, doi:https://doi.org/10.1016/j.aim.2014.03.011.

[7] Perry Hart. Mechanized study of coherent 2-groups, 2025. URL: https://github.com/PHart3/2-groups-agda/.

[8] Niles Johnson and Donald Yau. *2-Dimensional Categories*. Oxford University Press, 01 2021. doi:10.1093/oso/9780198871378.001.0001.

[9] Daniel R. Licata and Eric Finster. Eilenberg-MacLane spaces in homotopy type theory. In *Proceedings of the 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, New York, NY, USA, 2014. Association for Computing Machinery. doi:10.1145/2603088.2603153.

[10] nLab authors. bigroupoid. https://ncatlab.org/nlab/show/bigroupoid, May 2025. Revision 5.

[11] Egbert Rijke, Elisabeth Stenholm, Jonathan Prieto-Cubides, Fredrik Bakke, and others. The agda-unimath library. URL: https://github.com/UniMath/agda-unimath/.