.py file
.tex file
.pdf file
.lean file

Three Whitehead Theorems and Three Puppe Sequences

IntCat	D(∞-Cat)	$\vec{\Omega}$	$\vec{\mathrm{P}}$	$\vec{\mathrm{B}}$	$\vec{\mathrm{E}}$	IntShf	D(∞-Cat/C)	$\vec{\omega}$	$\vec{\mathrm{p}}$	$\vec{\mathrm{b}}$	$\vec{\mathrm{e}}$
IntGrpd	D(∞-Grpd)	$\overleftrightarrow{\Omega}$	$\overleftrightarrow{\mathrm{P}}$	$\overleftrightarrow{\mathrm{B}}$	$\overleftrightarrow{\mathrm{E}}$	IntAct	D(∞-Grpd/G)	$\overleftrightarrow{\omega}$	$\overleftrightarrow{\mathrm{p}}$	$\overleftrightarrow{\mathrm{b}}$	$\overleftrightarrow{\mathrm{e}}$
IntGrp	D(∞-Grpd$_0$)	Ω	P	B	E	IntAct$_0$	D(∞-Grpd$_0$/G$_0$)	ω	p	b	e

$$\forall(\mathrm{C{:}D}(\infty\text{-Cat})),\forall(\mathrm{D{:}D}(\infty\text{-Cat})),\forall(\mathrm{F{:}C} \longrightarrow \mathrm{D}),\forall(\mathrm{G{:}C} \longrightarrow \mathrm{D}),(\forall(\mathrm{n{:}Nat}),(\vec{\pi}_n\ \mathrm{F} = \vec{\pi}_n\ \mathrm{G})) \to \mathrm{F} = \mathrm{G}$$

$$\forall(\mathrm{X{:}D}(\infty\text{-Grpd})),\forall(\mathrm{Y{:}D}(\infty\text{-Grpd})),\forall(\mathrm{f{:}X} \longrightarrow \mathrm{Y}),\forall(\mathrm{g{:}X} \longrightarrow \mathrm{Y}),(\forall(\mathrm{n{:}Nat}),(\overleftrightarrow{\pi}_n\ \mathrm{f} = \overleftrightarrow{\pi}_n\ \mathrm{g})) \to \mathrm{f} = \mathrm{g}$$

$$\forall(\mathrm{X{:}D}(\infty\text{-Grpd}_0)),\forall(\mathrm{Y{:}D}(\infty\text{-Grpd}_0)),\forall(\mathrm{f{:}X} \longrightarrow \mathrm{Y}),\forall(\mathrm{g{:}X} \longrightarrow \mathrm{Y}),(\forall(\mathrm{n{:}Nat}),(\pi_n\ \mathrm{f} = \pi_n\ \mathrm{g})) \to \mathrm{f} = \mathrm{g}$$

$$\cdots \longrightarrow \vec{\pi}_1.\mathrm{obj\ C} \longrightarrow \vec{\pi}_1.\mathrm{obj\ D} \circlearrowright \vec{\pi}_0.\mathrm{obj}\ ((\mathbb{1}\ \mathrm{C}) \bullet ((\vec{\omega}.\mathrm{hom}\ (\mathbb{1}\ \mathrm{D})).\mathrm{hom\ f})\ ) \longrightarrow (\vec{\pi}_0.\mathrm{obj\ C}) \longrightarrow (\vec{\pi}_0.\mathrm{obj\ D})$$

$$\cdots \longrightarrow \overleftrightarrow{\pi}_1.\mathrm{obj\ E} \longrightarrow \overleftrightarrow{\pi}_1.\mathrm{obj\ B} \circlearrowright \overleftrightarrow{\pi}_0.\mathrm{obj}\ ((\mathbb{1}\ \mathrm{B}) \bullet ((\overleftrightarrow{\omega}.\mathrm{hom}\ (\mathbb{1}\ \mathrm{C})).\mathrm{hom\ f})\ ) \longrightarrow (\overleftrightarrow{\pi}_0.\mathrm{obj\ E}) \longrightarrow (\overleftrightarrow{\pi}_0.\mathrm{obj\ B})$$

$$\cdots \longrightarrow \pi_1.\mathrm{obj\ E}_0 \longrightarrow \pi_1.\mathrm{obj\ B}_0 \longrightarrow \pi_0.\mathrm{obj}\ ((\mathbb{1}\ \mathrm{B}_0) \bullet ((\omega.\mathrm{hom}\ (\mathbb{1}\ \mathrm{B}_0)).\mathrm{hom\ f})) \longrightarrow \pi_0.\mathrm{obj}\ (\mathrm{E}_0) \longrightarrow \pi_0.\mathrm{obj}(\mathrm{B}_0)$$

E. Dean Young and Jiazhen Xia

Plans to prove three variations of the Whitehead theorem and the exactness of three variations of the Puppe sequence of homotopy groups for based ∞-groupoids in Lean 4, with extensive use of Mathlib's categories, functors, natural transformations, Eilenberg-Moore theory, pullbacks, products, and material on simplicial sets

Copyright © October 19th 2023 Elliot Dean Young. All rights reserved.

We wish to acknowledge the collaborative efforts of E. Dean Young and Jiazhen Xia (⍰⍰⍰). The first author initially formulated the extensive outline and twelve goals presented in this research, and both made valuable contributions refining, extending, and implementing them.

1. Categories (see Mathlib's `Category X`; these can be bundled into `category`)
2. Functors (see Mathlib's `Functor C D`; these can be bundled into `functor`)
3. NaturalTransformation (see Mathlib's `NaturalTransform F G`; these can be bundled into `natural_transform`)
4. Equations (see Mathlib's `NatExt`; these are related to our `equation`)

1. Dom :
2. Cod :
3. Idn :
4. Id_1 :
5. Id_2 :
6. Ass :

1.

1. IntCat : Cat $\rightarrow$ Cat
2. IntShf : (X : Cat) $\rightarrow$ (C : (IntCat X)) $\rightarrow$ Cat
3. IntGrpd : Cat $\rightarrow$ Cat
4. IntAct : (X : Cat) $\rightarrow$ (G : (IntGrpd X)) $\rightarrow$ Cat
5. IntGrp : Cat $\rightarrow$ Cat
6. IntAct_0 : (X : Cat) $\rightarrow$ (G_0 : (IntGrp X)) $\rightarrow$ Cat

Below is a complete list of the symbols we define and the theorems we define and prove:

1. Contents

REP for ∞-groupoids	The replacement functor for ∞-groupoids
HEP for ∞-groupoids	The homotopy extension property
The Whitehead theorem	A map `F : D(∞-Grpd).Hom E B` is determined by λ`(n:Nat)`,$\overleftrightarrow{\pi}_n$ `F`.
`Chapter 9:` Internal Groupoids and their Internal Sheaves	
`IntGrpd Γ`	The category of internal groupoids in Γ
`IntAct Γ G`	The category of internal G-actions in Γ
The internal groupoid principal	`f ×_(B) f` forms an internal groupoid
The internal groupoid action principal	`f ×_(B) g` forms an internal groupoid action
$\overleftrightarrow{\mathrm{P}}$	The (remembrant derived) path space functor
$\overleftrightarrow{\mathrm{p}}\ \mathrm{C}$	The (remembrant derived) homotopy pullback functor
`Chapter 10:` The Puppe Sequence for ∞-Groupoids	
The Puppe sequence	$\cdots \longrightarrow \overleftrightarrow{\pi}_1(\mathrm{E}) \longrightarrow \overleftrightarrow{\pi}_1(\mathrm{B}) \circlearrowright \overleftrightarrow{\pi}_0(\overleftrightarrow{\omega}\,(\mathbb{1}\ \mathrm{C})\ \mathrm{f}) \longrightarrow \overleftrightarrow{\pi}_0(\mathrm{E}) \longrightarrow \overleftrightarrow{\pi}_0(\mathrm{B})$
`Chapter 11:` The Groupoid Fixed Point Principals	
The internal groupoid fixed point principal	`D(∞-Grpd)` is internal categories in itself
The internal groupoid action fixed point principal	`D(∞-Grpd/C)` is internal groupoid actions in itself
`PART III:` Based Connected ∞-Groupoids	
`Chapter 12:` Based Connected ∞-Groupoids	
$\mathrm{D}(\infty\text{-Grpd}_0)$	The derived category of based connected ∞-groupoids
$\mathrm{D}(\infty\text{-Grpd}_0/\mathrm{X}_0)$	The derived category of based connected ∞-groupoids over X_0.
$\Omega : \infty\text{-Grpd}_0 \longrightarrow \infty\text{-Grpd}$	The loop space functor
$\omega\ \mathrm{f} : \mathrm{D}(\infty\text{-Grpd}/\mathrm{D}_0) \longrightarrow \mathrm{D}(\infty\text{-Grpd}/\mathrm{C}_0)$	The homotopy fiber
$\pi_n : \infty\text{-Grpd}_0 \longrightarrow \mathrm{Set}$	The connected components functors
`Chapter 13:` The Whitehead Theorem for Based Connected ∞-Groupoids	
REP for based connected ∞-groupoids	The replacement functor on ∞-`Grpd`$_0$
HEP for based connected ∞-groupoids	The homotopy extension property for ∞-`Grpd`$_0$
Whitehead theorem (c)	A map `F : D(∞-Grpd`$_0$`).Hom E`$_0$ `B`$_0$ is determined by λ`(n:Nat)`,π_n `F`.
`Chapter 14:` Internal Groups	
`IntGrp Γ`	The category of internal groups in Γ
`IntAct`$_0$ `Γ G`$_0$	The category of internal G_0-actions in Γ
The internal group principal	Ω X forms an internal group in $\mathrm{D}(\infty\text{-Grpd})$
The internal group action principal	ω f forms an internal group action in $\mathrm{D}(\infty\text{-Grpd}/\mathrm{G}_0)$
P	The (remembrant derived) path space functor
p G_0	The (remembrant derived) homotopy fiber
`Chapter 15:` The Puppe Sequence for Based Connected ∞-Groupoids	
The Puppe sequence	$\cdots \longrightarrow \pi_1(\mathrm{E}_0) \longrightarrow \pi_1(\mathrm{B}_0) \longrightarrow \pi_0(\omega\ (\mathbb{1}\ \mathrm{X}_0)) \longrightarrow \pi_0(\mathrm{E}_0) \longrightarrow \pi_0(\mathrm{B}_0)$
`Chapter 16:` The Categorical Equivalences for Based Connected ∞-Groupoids	
The internal group fixed point principal	`D(∞-Grpd`$_0$`)` and internal groups in based spaces
The internal group action fixed point principal	`D(∞-Grpd`$_0$`/G`$_0$`)` and internal $\Omega\mathrm{G}_0$-actions in `D(∞-Grpd`$_0$`/G`$_0$`)`

2. Introduction

The main goal of this text is to prove the Whitehead theorem concerning based connected CW-complexes for the case of Mathlib's homotopy category of simplicial sets with a lifting condition. The book "Galois theories" by Borceux and Janelidze deserves special mention as an inspiration for the present project. That book details how to think about Galois theory using internal groupoids, internal G-presheaves, monadicity, comonadicity, and the constructions involved in Eilenberg Moore theory.

Here are the three Whitehead Theorems which form our main three goals:

(a) (The Whitehead theorem for ∞-categories) $\forall$(E:D(∞-Cat)),$\forall$(B:D(∞-Cat)),$\forall$(F:E $\longrightarrow$ B),$\forall$(G:E $\longrightarrow$ B),($\forall$(n:Nat),($\vec{\pi}_n$ F = $\vec{\pi}_n$ G)) $\to$ F = G, where $\vec{\pi}_n$ is notation for $\vec{\pi}$ n.

(b) (The Whitehead theorem for ∞-groupoids) $\forall$(E:D(∞-Grpd)),$\forall$(B:D(∞-Grpd)),$\forall$(F:E $\longrightarrow$ B),$\forall$(G:E $\longrightarrow$ B),($\forall$(n:Nat),($\overset{\leftrightarrow}{\pi}_n$ F = $\overset{\leftrightarrow}{\pi}_n$ G)) $\to$ F = G, where $\overset{\leftrightarrow}{\pi}_n$ is notation for $\overset{\leftrightarrow}{\pi}$ n.

(c) (The Whitehead theorem for based connected ∞-groupoids) $\forall$(E:D(∞-Grpd$_0$)),$\forall$(B:D(∞-Grpd$_0$)),$\forall$(f:E $\longrightarrow$ B),$\forall$(G:E $\longrightarrow$ B),($\forall$(n:Nat),(π_n F = π_n G)) $\to$ F = G, where π_n is notation for π n.

We will be using two different models of ∞-Grpd, a model of ∞-Cat, a model of D(∞-Cat) (based on REP and HEP detailed in chapter 3), and model of ⍰(∞-Cat) in which Whitehead theorem (a) is forced. This is perhaps unimpressive, and also leads us to thinking about replacement as an endofunctor `repl : Functor∞-Cat ∞-Cat`, which is involved in both the statement and proof of REP, and which is designed in a way such that HEP holds.

We will use the following models in the theorem above:

(i) ∞-Cat is the category of quasicategories.

(ii) ∞-Grpd is the category of simplicial sets with the Kan lifting condition.

(iii) ∞-Grpd$_0$ is the category of based connected simplicial sets with the Kan lifting condition.

The operations require great care in their definition. The existence of a base point makes π_n relatively easy to define, while $\vec{\pi}_n$ and $\overset{\leftrightarrow}{\pi}_n$ require careful consideration and get much larger by comparison:

(i) $\vec{\pi}_n : \infty\text{-Cat} \longrightarrow \text{Set}$

(ii) $\overleftrightarrow{\pi}_n : \infty\text{-Grpd} \longrightarrow \text{Set}$

(iii) $\pi_n : \infty\text{-Grpd}_0 \longrightarrow \text{Set}$

we also form

(i) $\mathrm{D}(\vec{\pi}_n) : \mathrm{D}(\infty\text{-Cat}) \longrightarrow \mathrm{D}(\text{Set}) \simeq \text{Set}$

(ii) $\mathrm{D}(\overleftrightarrow{\pi}_n) : \mathrm{D}(\infty\text{-Grpd}) \longrightarrow \mathrm{D}(\text{Set}) \simeq \text{Set}$

(iii) $\mathrm{D}(\pi_n) : \mathrm{D}(\infty\text{-Grpd}_0) \longrightarrow \mathrm{D}(\text{Set}) \simeq \text{Set}$

and

1. $\vec{\Omega} : \infty\text{-Cat} \longrightarrow \infty\text{-Cat}$ is the internal hom functor $[\Delta^1,\text{-}]$ (directed path space)
2. $\overleftrightarrow{\Omega} : \infty\text{-Grpd} \longrightarrow \infty\text{-Grpd}$ is the internal hom functor [I,-] (path space)
3. Ω is the loop space functor

The first of the Whitehead theorems, (a), is the most abstract. The third, (c), is the one from Whitehead's original papers. The second forms a nice intermediate between the two.

The choice of quasicategories gives nice integration with Mathlib's existing features (though technically only the inner horns and simplices are defined, not even the category of quasicategories itself), a possible benefit over a more "synthetic" approach based on forcing the three Whitehead theorems and three Puppe sequences at the outset (along with the functors, natural isomorphisms, and equations in the first several pages).

The main technical feature in the proofs of these theorems concerns a lifting property which successively lifts a homotopy*** along a single attachment of Δ^n along its boundary $\partial\Delta^n$. A homotopy $h : \partial\Delta^n \times \Delta^1 \longrightarrow Y$ between $f, g : \partial\Delta^n \longrightarrow Y$ extends to a map $H : \Delta^n \times \Delta^1 \longrightarrow Y$. H(-,1) and g match on $\partial\Delta^n$, producing a map $f : X \longrightarrow Y$, where X consists of two copies of Δ^n glued together at the boundary. Consider a space X' formed as a quotient of $\Delta^n \times \Delta^1$ by $\partial\Delta^n \times \Delta^1$. There is a map $\phi : X \longrightarrow X'$. An induction hypothesis on f and g involving π_n ensures that the aparent map $X \longrightarrow Y$ lifts along ϕ, producing a map from $\Delta^n \times \Delta^1$ which is constant on $\partial\Delta^n \times \Delta^1$. Stacking this on top of H can be done using an isomorphism between Δ^1 and Δ^1 glued with itself along different endpoints. Altogether this produces a homotopy between f and g.

*** Note that a homotopy here is to do with the directed derived category of an overcategory $\mathrm{D}(\infty\text{-Cat}/C)$, and it consists of really a sequence of compatible directed homotopies with the odd morphisms formed from reversed copies of Δ^1. Really we have two such categories, one of which consists of formal words, and another which involves ∞-categories and ∞-functors in the image of `repl`)

Using the mentioned categories, we will define three different kinds of derived category:

1. $D(\infty\text{-Cat}) : \text{Cat}$ (the directed derived category of ∞-categories)
2. $D(\infty\text{-Grpd}) : \text{Cat}$ (the derived category of ∞-groupoids)
3. $D(\infty\text{-Grpd}_0) : \text{Cat}$ (the derived category of based ∞-groupoids)

These are formed by identifying those morphisms (∞-functors) between which there is a natural transformation.

We also create a second kind of category, one for each of the objects in the respective categories above:

1. For $C : D(\infty\text{-Cat})$, a category $D(\infty\text{-Cat}/C) : \text{Cat}$
2. For $G : D(\infty\text{-Grpd})$, a category $D(\infty\text{-Grpd}/G) : \text{Cat}$
3. For $G_0 : D(\infty\text{-Grpd}_0)$, a category $D(\infty\text{-Grpd}_0/G_0) : \text{Cat}$

For the model built on simplicial sets, $\vec{\Omega}$ will be representable by Δ^1 with respect to an internal hom, and $\overleftrightarrow{\Omega}$ will be representable by a model of the unit interval $I := [0,1]$.

The six mentioned internal structures are related to six functors:

(I) $\vec{\Omega} : \infty\text{-Cat} \longrightarrow \infty\text{-Cat}$ (notation for the directed path space functor, related to $[\Delta^1,\text{-}]$). $D(\vec{\Omega})$ factors through internal categories in $D(\infty\text{-Cat})$ by a categorical equivalence $D(\infty\text{-Cat}) \cong \text{IntCat}\ D(\infty\text{-Cat})$ (internal categories in $D(\infty\text{-Cat})$)

(II) $\vec{\omega}\,(\mathbb{1}\ C) : \infty\text{-Cat}/C \longrightarrow \infty\text{-Cat}/C$, the derived directed homotopy pullback with $\mathbb{1}\ C$. $D(\vec{\omega}\,(\mathbb{1}\ C))$ factors through a categorical equivalence between $D(\infty\text{-Cat}/C)$ and internal $\vec{P}C$-presheaves in $D(\infty\text{-Cat}/C)$.

(III) $\overleftrightarrow{\Omega} : \infty\text{-Grpd} \longrightarrow \infty\text{-Grpd}$ (notation for the path space functor $[I,\text{-}]$), the derived homotopy pullback of an ∞-groupoid with itself. $D(\overleftrightarrow{\Omega})$ factors through a categorical equivalence between $D(\infty\text{-Grpd})$ and internal groupoids in $D(\infty\text{-Grpd})$

(IV) $\overleftrightarrow{\omega}\,(\mathbb{1}\ X) : \infty\text{-Grpd}/X \longrightarrow \infty\text{-Grpd}/X$, the derived homotopy pullback with $\mathbb{1}\ X$. $D(\overleftrightarrow{\omega}\,(\mathbb{1}\ X))$ factors through internal $\overleftrightarrow{P}X$

(V) $\Omega : \infty\text{-Grpd}_0 \longrightarrow \infty\text{-Grpd}_0$, the loop space functor. $D(\Omega)$ factors through a categorical equivalence between $D(\infty\text{-Grpd}_0)$ and internal groups in $D(\infty\text{-Grpd}_0)$ (the loop space functor on connected based ∞-groupoids)

(VI) $\omega\,(\mathbb{1}\ \mathrm{X}) : \infty\text{-}\mathrm{Grpd}_0/\mathrm{X}_0 \longrightarrow \infty\text{-}\mathrm{Grpd}_0/\mathrm{X}_0$, the homotopy pullback with the base of X_0. $\mathrm{D}(\omega\ (\mathbb{1}\ \mathrm{X}))$ factors through internal PX_0-actions in based connected spaces over X_0.

(v) in the above is shown here and (vi) in the above is shown in a typical treatment of G-principal bundles.

The functors $\vec{\omega}\ (\mathbb{1}\ \mathrm{C})$, $\overleftrightarrow{\omega}\ (\mathbb{1}\ \mathrm{X})$, and $\omega\ (\mathbb{1}\ \mathrm{C})$ in the above ensue from a more general construction:

1. For $\mathrm{C}, \mathrm{D} : \mathrm{D}(\infty\text{-}\mathrm{Cat})$, and $\mathrm{f} : \mathrm{C} \longrightarrow \mathrm{D}$, $\vec{\omega}\,\mathrm{f} : \mathrm{D}(\infty\text{-}\mathrm{Cat}/\mathrm{D}) \longrightarrow \mathrm{D}(\infty\text{-}\mathrm{Cat}/\mathrm{C})$ (derived directed homotopy pullback)
2. For $\mathrm{B}, \mathrm{E} : \mathrm{D}(\infty\text{-}\mathrm{Grpd})$, and $\mathrm{f} : \mathrm{E} \longrightarrow \mathrm{B}$, $\overleftrightarrow{\omega}\ \mathrm{f} : \mathrm{D}(\infty\text{-}\mathrm{Grpd}/\mathrm{B}) \longrightarrow \mathrm{D}(\infty\text{-}\mathrm{Grpd}/\mathrm{E})$ (derived homotopy pullback)
3. For $\mathrm{B}_0, \mathrm{E}_0 : \mathrm{D}(\infty\text{-}\mathrm{Grpd}_0)$, and $\mathrm{f} : \mathrm{E}_0 \longrightarrow \mathrm{B}_0$, $\omega\ \mathrm{f} : \mathrm{D}(\infty\text{-}\mathrm{Grpd}_0/\mathrm{B}_0) \longrightarrow \mathrm{D}(\infty\text{-}\mathrm{Grpd}_0/\mathrm{E}_0)$ (homotopy pullback with the base)

These functors produce what you might call "fixed points" of the "internal X in" operation defined on categories with pullbacks (or products in the case of the third Whitehead theorem). We obtain six categorical equivalences witnessed by these constructions. To see how, we first name the six mentioned functors I - VI in the above:

1. The directed path space, the path space, and loop space form components of the functors $\vec{\mathrm{P}}$, $\overleftrightarrow{\mathrm{P}}$, and P, which are valued in internal categories, internal groupoids, and internal groups respectively.
   (a) $\vec{\mathrm{P}} : \mathrm{D}(\infty\text{-}\mathrm{Cat}) \longrightarrow \mathrm{Cat}\ \mathrm{D}(\infty\text{-}\mathrm{Cat})$
   (b) $\overleftrightarrow{\mathrm{P}} : \mathrm{D}(\infty\text{-}\mathrm{Grpd}) \longrightarrow \mathrm{Grpd}\ \mathrm{D}(\infty\text{-}\mathrm{Grpd})$
   (c) $\mathrm{P} : \mathrm{D}(\infty\text{-}\mathrm{Grpd}_0) \longrightarrow \mathrm{Grp}\ \mathrm{D}(\infty\text{-}\mathrm{Grpd})$ (see here)
2. The directed homotopy pullback, the homotopy pullback, and the homotopy pullback with the base form components of the functors $\mathrm{Alg}(\mathrm{Mon}(\vec{\omega}))$, $\mathrm{Alg}(\mathrm{Mon}(\overleftrightarrow{\omega}))$, and Alg(Mon(p)), respectively.
   (a) $\vec{\mathrm{p}}\,(\mathbb{1}\ \mathrm{C}) : \mathrm{D}(\infty\text{-}\mathrm{Cat}/\mathrm{C}) \longrightarrow \mathrm{IntShf}\ \mathrm{D}(\infty\text{-}\mathrm{Cat}/\mathrm{C})\ \vec{\Omega}\mathrm{C}$
   (b) $\overleftrightarrow{\mathrm{p}}\,(\mathbb{1}\ \mathrm{X}) : \mathrm{D}(\infty\text{-}\mathrm{Grpd}/\mathrm{X}) \longrightarrow \mathrm{IntAct}\ \mathrm{D}(\infty\text{-}\mathrm{Grpd}/\mathrm{X})\ \overleftrightarrow{\Omega}\mathrm{X}$
   (c) $\mathrm{p}\,(\mathbb{1}\ \mathrm{X}_0) : \mathrm{D}(\infty\text{-}\mathrm{Grpd}_0/\mathrm{X}_0) \longrightarrow \mathrm{IntAct}_0\ \mathrm{D}(\infty\text{-}\mathrm{Grpd}_0/\mathrm{X}_0)\ \Omega\mathrm{X}_0$

Above, the functors $\vec{\mathrm{P}}$, $\overleftrightarrow{\mathrm{P}}$, P, $\vec{\mathrm{p}}$, $\overleftrightarrow{\mathrm{p}}$, and p can be formed out of $\vec{\Omega}$, $\overleftrightarrow{\Omega}$, Ω, $\vec{\omega}$, $\overleftrightarrow{\omega}$, and ω using constructions from Eilenberg-Moore theory.

These six new functors combine with the functors below to form categorical equivalences:

1. The directed homotopy colimit of a point with an internal category in $\mathrm{D}(\infty\text{-Cat})$ as a diagram, the homotopy colimit of a constant functor with an internal internal group as a diagram
   (a) $\vec{\mathrm{B}} : \mathrm{IntCat}\ \mathrm{D}(\infty\text{-Cat}) \longrightarrow \mathrm{D}(\infty\text{-Cat})$
   (b) $\overleftrightarrow{\mathrm{B}} : \mathrm{IntGrpd}\ \mathrm{D}(\infty\text{-Grpd}) \longrightarrow \mathrm{D}(\infty\text{-Grpd})$
   (c) $\mathrm{B} : \mathrm{IntGrpd}\ \mathrm{D}(\infty\text{-Grpd}_0) \longrightarrow \mathrm{D}(\infty\text{-Grpd}_0)$ (see here)
2. The clutching functors are inverse to the above functors up to natural isomorphism:
   (a) $\vec{\mathrm{b}} : \mathrm{IntShf}\ \mathrm{D}(\infty\text{-Cat}/\mathrm{C})\ \vec{\Omega}\mathrm{C} \longrightarrow \mathrm{D}(\infty\text{-Cat}/\mathrm{C})$
   (b) $\overleftrightarrow{\mathrm{b}} : \mathrm{IntAct}\ \mathrm{D}(\infty\text{-Cat}/\mathrm{C})\ \overleftrightarrow{\Omega}\mathrm{X} \longrightarrow \mathrm{D}(\infty\text{-Cat}/\mathrm{C})$
   (c) $\mathrm{b} : \mathrm{IntAct}_0\ \mathrm{D}(\infty\text{-Grpd}_0/\mathrm{X}_0)\ \Omega\mathrm{X}_0 \longrightarrow \mathrm{D}(\infty\text{-Grpd}_0/\mathrm{X}_0)$

We will show six categorical equivalences featuring these:

1. $\vec{\mathrm{P}} \bullet \vec{\mathrm{B}} \cong \mathbb{1}$ (IntCat $\mathrm{D}(\infty\text{-Cat})$) and $\vec{\mathrm{B}} \bullet \vec{\mathrm{P}} \cong \mathbb{1}\ \mathrm{D}(\infty\text{-Cat})$
2. $\overleftrightarrow{\mathrm{P}} \bullet \overleftrightarrow{\mathrm{B}} \cong \mathbb{1}$ (IntGrpd $\mathrm{D}(\infty\text{-Grpd})$) and $\overleftrightarrow{\mathrm{B}} \bullet \overleftrightarrow{\mathrm{P}} \cong \mathbb{1}\ \mathrm{D}(\infty\text{-Grpd})$
3. $\mathrm{P} \bullet \mathrm{B} \cong \mathbb{1}$ (IntGrp $\mathrm{D}(\infty\text{-Grpd}_0)$) and $\mathrm{B} \bullet \mathrm{P} \cong \mathbb{1}\ \mathrm{D}(\infty\text{-Grpd}_0)$ (see here)
4. $\vec{\mathrm{p}} \bullet \vec{\mathrm{b}} \cong \mathbb{1}$ (IntShf $\mathrm{D}(\infty\text{-Cat}/\mathrm{C})\ \vec{\mathrm{P}}\mathrm{C}$) and $\vec{\mathrm{b}} \bullet \vec{\mathrm{p}} \cong \mathbb{1}\ \mathrm{D}(\infty\text{-Cat}/\mathrm{C})$
5. $\overleftrightarrow{\mathrm{p}} \bullet \overleftrightarrow{\mathrm{b}} \cong \mathbb{1}$ (IntAct $\mathrm{D}(\infty\text{-Cat}/\mathrm{C})\ \overleftrightarrow{\mathrm{P}}\mathrm{X}$) and $\overleftrightarrow{\mathrm{b}} \bullet \overleftrightarrow{\mathrm{p}} \cong \mathbb{1}\ \mathrm{D}(\infty\text{-Cat}/\mathrm{C})$
6. $\mathrm{p} \bullet \mathrm{b} \cong \mathbb{1}$ ($\mathrm{IntAct}_0\ \mathrm{D}(\infty\text{-Grpd}_0/\mathrm{X}_0)\ \mathrm{PX}_0$) and $\mathrm{b} \bullet \mathrm{p} \cong \mathbb{1}\ \mathrm{D}(\infty\text{-Grpd}_0/\mathrm{X}_0)$ (see here)

We will make extensive use of Mathlib's bicategory of categories, as well as Mathlib's bicategories in general. We further use Mathlib's pullbacks and categorical products, as well as their Eilenberg-Moore theory. I'd like to extend my appreciation to Scott Morison and all the contributors who have put their efforts into creating a great category theory section in Mathlib.

Altogether, the project gets the following "periodic table" of 24 functors featured on the front cover:

`D(∞-Cat)`	$\vec{\Omega}$	$\vec{\mathrm{P}}$	$\vec{\mathrm{B}}$	$\vec{\mathrm{E}}$	`D(∞-Cat/C)`	$\vec{\omega}$	$\vec{\mathrm{b}}$	$\vec{\mathrm{p}}$	$\vec{\mathrm{e}}$
`D(∞-Grpd)`	$\overleftrightarrow{\Omega}$	$\overleftrightarrow{\mathrm{P}}$	$\overleftrightarrow{\mathrm{B}}$	$\overleftrightarrow{\mathrm{E}}$	`D(∞-Grpd/G)`	$\overleftrightarrow{\omega}$	$\overleftrightarrow{\mathrm{b}}$	$\overleftrightarrow{\mathrm{p}}$	$\overleftrightarrow{\mathrm{e}}$
$\mathtt{D(\infty\text{-}Grpd_0)}$	Ω	P	B	E	$\mathtt{D(\infty\text{-}Grpd_0/G_0)}$	ω	b	p	e

Here are the names of the symbols featured above:

`Deductive`	`Remembrant`	`Classifying`
$\vec{\Omega}$ (Directed path space)	$\vec{\mathrm{P}}$ (Remembrant derived directed path space)	$\vec{\mathrm{B}}$ (Classifying space for internal categories)
$\overleftrightarrow{\Omega}$ (Path space)	$\overleftrightarrow{\mathrm{P}}$ (Remembrant derived path space)	$\overleftrightarrow{\mathrm{B}}$ (Classifying space for internal groupoids)
Ω (Loop space)	P (Remembrant derived loop space)	B (Classifying space for internal groups)
$\vec{\omega}$ (Directed homotopy pullback)	$\vec{\mathrm{p}}$ (Remembrant derived directed homotopy pullback)	$\vec{\mathrm{b}}$ (Classifying space for internal presheaves)
$\overleftrightarrow{\omega}$ (Homotopy pullback)	$\overleftrightarrow{\mathrm{p}}$ (Remembrant derived homotopy pullback)	$\overleftrightarrow{\mathrm{b}}$ (Classifying space for internal groupoid actions)
ω (Homotopy fiber)	p (Remembrant derived homotopy fiber)	b (Classifying space for internal group actions)

The term "remembrant" in the above is not common terminology. It is intended to mean that these functors "remember" all of the structure which one can put on the derived functors of the functors in the left collumn . In other words, the second collumn features functors which are valued in categories of internal objects wheras the left collumn forms particular components of those structures.

The notation here is both an attempt to make the various analogies manifest while sticking to familiar notation where available (such as in the case of Ω and B, which match the ordinary usage of these symbols). In the above, P could be said to stand for "path space" and p for "pullback", while at the same time this matches a nice theme that our capitcal letters reflect various internal structures and their lower-case forms reflect the corresponding actions.

The mentioned "fixed point principals", which identify categories which are internal X's in themselves, form important consequences of the three Whitehead theorems. All in all, there are twelve important theorems we will show:

Twelve Goals

(I) Define and inhabit the `whitehead_theorem_for_categories : Type`.

(II) Define the Puppe sequence for ∞-categories and prove its exactness.

(III) Define and inhabit the `internal_category_fixed_point_principal : Type`.

(IV) Define and inhabit the `internal_sheaf_fixed_point_principal : Type`.

(V) Define and inhabit the `whitehead_theorem_for_groupoids : Type`.

(VI) Define the Puppe sequence for ∞-groupoids and prove its exactness

(VII) Define and inhabit the `internal_groupoid_fixed_point_principal : Type`.

(VIII) Define and inhabit the `internal_groupoid_action_fixed_point_principal : Type`.

(IX) Define and inhabit the `whitehead_theorem : Type`.

(X) Define the Puppe sequence for based connected ∞-groupoids and prove its exactness

(XI) Define and inhabit the `internal_group_categorical_equivalence : Type`.

(XII) Define and inhabit the `internal_group_action_categorical_equivalence : Type`.

Note that the last two are categorical equivalences and not fixed points of ”internal X in”. Despite not making a perfect analogy, the three-fold nature of the table (producing three Whitehead theorems using Lean’s `Nat` and three Puppe sequences)

None of these theorems are currently contained in Mathlib. The last four are famously known as:

1. The Whitehead theorem
2. The Puppe sequence and its exactness
3. The theorem that $\Omega \bullet B \cong \mathbb{1}$ (IntGrp D(∞-Grpd)) and $B \bullet \Omega \cong \mathbb{1}$ D(∞-Grpd$_0$)
4. The theorem that BG classifies G-principal bundles

3. Introduction to Lean 4

The main way to tell Lean 4 what something means is with `def`, which defines a term in dependent type theory. Much in the same way as other computer languages, we then supply the type of the term (e.g. `Int` for integer), followed by the formula itself:

Lean 1

```
def zero : Nat := 0
```

Here we have introduced an natural number `n` using the type `Nat` that comes with Lean 4. In addition to defining terms, Lean allows you to perform computations and prove theorems.

As a beginner, it's normal to take some time to get comfortable with Lean and formal proof systems. It's a journey that requires practice and patience. Lean has an active community that provides support and resources to help you along the way.

Constituents of `x, y : X` of types `X` can also stand to be equal or unequal, written `x = y`, and it is the properties of equality which in addition to the dependent type theory make a type behave like a set. Equality satisfies the three properties of an equivalence relation, which we cover presently. Consider first the reflexivity property of equality:

Lean 2

```
def reflexivity {X : Type} {x : X} : x = x := Eq.refl
↪ x
```

This command defines a function called reflexivity that proves the reflexivity property of equality. The function takes two type parameters: `X` represents the type of the elements being compared, and x represents an element of type X. It also takes an argument ω which is a proof that x is equal to itself (x = x). The function body states that the result of `reflexivity` is the proof ω itself using the `Eq.refl` constructor, which indicates that x is equal to itself.

In Lean 4, {x : X} represents an implicit argument, where Lean will attempt to infer the value of x based on the context. (x : X) represents an explicit argument, requiring the value of x to be provided explicitly when using the function or definition.

Lean 3

```
def symmetry {X : Type} {x : X} {y : X}  (p : x = y)
↪  := Eq.symm p
```

This command defines a function called symmetry that proves the symmetry property of equality. It takes three type parameters: X represents the type of the elements being compared, and x and y represent elements of type X. The function also takes an argument ω which is a proof that x is equal to y (x=y). The function body states that the result of `symmetry` is the proof ω itself using the `Eq.symm` constructor, which allows you to reverse an equality proof.

Lean 4

```
def transitivity {X : Type} {x : X} {y : X} {z : X}
↪  (p : x = y) (q : y = z) := Eq.trans ω q
```

This command defines a function called transitivity that proves the transitivity property of equality. It takes four type parameters: X represents the type of the elements being compared, and x, y, and z represent elements of type X. The function also takes two arguments ω and q. ω is a proof that x is equal to y (x = y), and q is a proof that y is equal to z (y = z). The function body states that the result of transitivity is the proof of the composition of ω and q using the `Eq.trans` constructor, which allows you to combine two equality proofs to obtain a new one.

These Lean commands define functions that prove fundamental properties of equality: reflexivity (every element is equal to itself), symmetry (equality is symmetric), and transitivity (equality is transitive). These properties are essential for reasoning about equality in mathematics and formal proofs.

We must also require that functions satisfy extensionality:

Lean 5

```
def extensionality (f g : X → Y) (p : (x:X) → f x =
↪  g x) : f = g := funext p
```

Extensionality, a key characteristic of sets and types, asserts that functions which are equal on all values are themselves equal, and it is featured prominently in what is perhaps the most well known mathematical foundations of ZFC.

There are several other features of equality with respect to functions which we should be aware of:

Lean 6

```
def equal_arguments {X : Type} {Y : Type} {a : X} {b
↪  : X} (f : X → Y) (p : a = b) : f a = f b :=
↪  congrArg f p

def equal_functions {X : Type} {Y : Type} {f₁ : X →
↪  Y} {f₂ : X → Y} (p : f₁ = f₂) (x : X) : f₁ x =
↪  f₂ x := congrFun ω x

def pairwise {A : Type} {B : Type} (a₁ : A) (a₂ : A)
↪  (b₁ : B) (b₂ : B) (p : a₁ = a₂) (q : b₁ = b₂) :
↪  (a₁,b₁)=(a₂,b₂) := (congr ((congrArg Prod.mk) p)
↪  q)
```

The tutorial here provides a good introduction to using the dependent type theory in Lean.

4. Unicode

Here is a list of the unicode characters we will use:

Symbol	Unicode	VSCode shortcut	Use
Lean's Kernel			
$\times$	2A2F	`\times`	Product of types
$\rightarrow$	2192	`\rightarrow`	Hom of types
$\langle , \rangle$	27E8,27E9	`\langle,\rangle`	Product term introduction
$->sto$	21A6	`\mapsto`	Hom term introduction
$\wedge$	2227	`\wedge`	Conjunction
$\vee$	2228	`\vee`	Disjunction
$\forall$	2200	`\forall`	Universal quantification
$\exists$	2203	`\exists`	Existential quantification
$\neg$	00AC	`\neg`	Negation
Variables and Constants			
a,b,c,...,z	1D52,1D56		Variables and constants
0,1,2,3,4,5,6,7,8,9	1D52,1D56		Variables and constants
⁻	207B		Variables and constants
$0,1,2,3,4,5,6,7,8,9$	2080 - 2089	`\0-\9`	Variables and constants
𝔸,...,ℤ	1D538		
𝕒,...,𝕫	1D552		
A,...,Z	1D41A		
a,...,z	1D41A		
α-ω,Α-Ω	03B1-03C9		Variables and constants
Categories			
𝟙	1D7D9	`\b1`	The identity morphism
$\circ$	2218	`\circ`	Composition
Bicategories			
$\bullet$	2022	`\smul`	Horizontal composition of objects
Adjunctions			
$\rightleftarrows$	21C4	`\rightleftarrows`	Adjunctions
$\leftrightarrows$	21C6	`\leftrightarrows`	Adjunctions
.	1BC94		Right adjoints
˙	0971		Left adjoints
$\dashv$	22A3	`\dashv`	The condition that two functors are adjoint
Monads and Comonads			
?,¿	003F, 00BF	?,\?	The corresponding (co)monad of an adjunction
!,¡	0021, 00A1	!, \!	The (co)-Eilenberg-(co)-Moore adjunction
ꜝ,ꜞ	A71D, A71E		The (co)exponential maps
Miscellaneous			
$\sim$	223C	`\sim`	Homotopies
$\simeq$	2243	`\equiv`	Equivalences
$\cong$	2245	`\cong`	Isomorphisms
$\perp$	22A5	`\bot`	The overobject classifier
∞	221E	`\infty`	Infinity categories and infinity groupoids
⃡	20D7		Homotopical operations on ∞-categories
⃗	20E1		Homotopical operations on ∞-groupoids

Of these, the characters $^{!}$, $^{¡}$, $_{.}$, $^{\cdot}$, $^{\rightarrow}$, and $^{\leftrightarrow}$ do not have VSCode shortcuts, and so we provide alternatives for them.

It is not possible to copy the from the pdf to the clipboard while preserving the integrity of the code. To see the official Lean 4 file please click the link on the top right of the front page or this.

Chapter 1: Mathlib's Category Theory Section

I intend for this to cover some of the basics of Mathlib's category theory in the final document. For now, just know that we'll be using bicategories, the bicategory of categories, Cat, Functor, NaturalTransform, its Eilenberg-Moore theory constructions, and its material on simplicial sets.

5. Mathlib's Category, Functor, NaturalTransform

Here I intend to cover some of the basics of Mathlib's Categories, Functors, and NaturalTransformations. I will for instance write out checks (`#check`) of the main types, as well as some instructions as to how to create new examples of these structures for beginners.

6. Mathlib's bicategories

Even though we are only going to use strict bicategories, strict bicategories work terribly with type systems and necessitate the use of `cast`. Lean has only ever used bicategories for this reason. Mathlib's bicategories are also enabled by `aesop`, which plays a nice role in inferring simple lemmas such as ones involving identity and associativity laws. The bicategory of categories is the only bicategory we'll really need, if at all.

7. Mathlib's adjunctions, monads, and comonads

We're going to make extensive use of adjunctions, specifically the ones involved in directed homotopy pullback, homotopy pullback, and homotopy kernel. These are each adjoint to a functor which post-composes with a morphism.

We will also make a lot of use of monads and comonads.

8. Mathlib's Eilenberg-Moore theory

Mathlib has great constructions of the fundamentals in Eilenberg-Moore theory. It turns out that we'll not actually use any co-Eilenberg co-Moore theory. We'll make use of:

1. The monad corresponding to an adjunction
2. The adjunction corresponding to a monad (Eilenberg-Moore adjunction)

9. Mathlib's pullbacks and products

Pullbacks and products are essential to the definition of the internal structures we're using. For our purposes, it makes the most sense to define pullbacks in a way which handles the assumption of their existence pleasently (and without much work for the user).

In the material that follows, we will be interested in forming pullbacks for four of the six internal structures that we make. We also take care to establish the definition of a fixed point of an endofunctor of Bicategory.Cat (Mathlib's category of categories).

10. Mathlib's simplicial sets

Mathlib already has simplicial sets, along with the the definition of the n-simplices Δ^n and the inner horns $\Lambda[;]$. We

1. Mathlib's definition of Δ^n

Lean 7

```
-- #check Δⁿ
/-
Goal: check Lean's n-simplices and
-/
```

2. Mathlib's definition of the inner horns

Lean 8

```
-- #check ???
```

3. Mathlib's definition of the inclusion of inner horns

Lean 9

```
-- #check ???
```

4. Defining the pushout of quasicategories

Lean 10

```
-- #check ???
```

5. Defining the boundary of Δ^n

Lean 11

```
-- #check ???
```

6. Pushouts in simplicial sets.

Lean 12

```
-- #check ???
```

7. A lemma involving the colimit of a diagram made out of subsimplicial sets, establishing on certain grounds that this colimit is isomorphic to the whole simplicial set.

Lean 13

```
-- #check ???
```

8. Defining the quasicategory lifting condition using Mathlib's predefined inner horns

Lean 14

```
-- #check ???
```

9. Defining the Kan lifting condition using Mathlib's predefined horns

Lean 15

```
-- #check ???
```

10. Lifting conditions and products

Lean 16

```
-- #check ???
```

11. The inner hom of simplicial sets (we need to see what Mathlib already has for this)

Lean 17

```
-- #check ???
```

12. Lifting conditions and the product of simplicial sets

Lean 18

```
-- #check ???
```

13. Lifting conditions and the path space $[\Delta^1, \text{-}]$

Lean 19

```
-- #check ???
```

14.

15. Lifting conditions and hom of simplicial sets

Lean 20

```
-- #check ???
```

16. Mathlib's cartesian closed category.

Lean 21

```
-- #check ???
```

17. Defining the cubes $C_n = (\Delta^1)^n$

Lean 22

```
-- #check ???
```

18. Defining the boundary of cubes C_n

Lean 23

```
-- #check ???
```

19. Defining the boundary of simplicial sets Δ^{n}

Lean 24

```
-- #check ???
```

PART 1: ∞-CATEGORIES

Chapter 2: ∞-Cat

This chapter and the next chapter are more technical and difficult than the rest of the book.

1. Defining D(∞-Cat) by formally inverting weak equivalences.
2. Defining D(∞-Cat/C) by formally inverting weak equivalences.
3. Defining a fibrant replacement functor for ∞-Cat
4. Defining a fibrant replacement functor for ∞-Cat/C
5. We first construct both the category D(∞-Cat) and, for each C : D(∞-Cat), the category D(∞-Cat/C) by formally inverting weak equivalences in the category of quasicategories and the category of quasicategories over C.

11. Ω

Our choice of symbols refects our choice of three variations of the Whitehead theorem and three Puppe sequences. $\vec{\Omega}$, the analogue of loop space, is the internal hom functor $[\Delta^1$,-$]$: ∞-Cat $\longrightarrow \infty$-Cat. This is not hard to construct, with the main lemma being that the path space of a quasicategory has the quasicategory lifting conditon.

We will be interested in one formal model of D(∞-Cat) which consists of formal compositions $f_1 \bullet g_1 \bullet f_2 \bullet g_2 \bullet$ ⍰ $\bullet f_n \bullet g_n$, where $g_n : Dom(f_{n+1}) \longrightarrow$??? is a weak equivalence, and something similar for D(∞-Cat). However, it is still vital to have the replacement functor `repl`, which ensures the Whitehead theorem for particular ∞-categories which are constructed out of attaching maps.

12. ω

$\vec{\Omega}$ is to internal categories as $\vec{\omega}$ is to internal C-presheaves. As such, $\vec{\omega}$ is a functor from ∞-Cat. It is also called directed homotopy pullback. These functors will later be used to produce functors $\vec{\mathrm{P}} : \mathrm{D}(\infty\text{-Cat}) \longrightarrow \mathrm{IntCat}\ \mathrm{D}(\infty\text{-Cat})$ and $\vec{\mathrm{p}} : \mathrm{D}(\infty\text{-Cat}/\mathrm{C}) \longrightarrow \mathrm{IntShf}\ (\vec{\mathrm{P}}\ \mathrm{C})\ \mathrm{D}(\infty\text{-Cat}/\mathrm{C})$.

13. π_n

Chapter 3: The Whitehead Theorem for ∞-Categories

In this chapter, we take on the objective of Whitehead theorem (a), out of which we will prove the other more concrete Whitehead theorems.

$$\forall(\mathrm{E{:}D}(\infty\text{-Cat})),\forall(\mathrm{B{:}D}(\infty\text{-Cat})),\forall(\mathrm{F{:}E} \longrightarrow \mathrm{B}),\forall(\mathrm{G{:}E} \longrightarrow \mathrm{B}),(\forall(\mathrm{n{:}Nat}),(\vec{\pi}_n\ \mathrm{F} = \vec{\pi}_n\ \mathrm{G})) \longrightarrow \mathrm{F} = \mathrm{G}$$

We can attempt to form a slightly different category, much like the above, called □(∞-Cat), at first, and in a formal way, so as to create a category whose object component □(∞-Cat).α matches the object component ∞-Cat.α while featuring the above theorem in a formal way. However, with this as our model of D(∞-Cat), we may then also be interested in the establishment of a model in which the Whitehead theorem is demonstrated, with the main idea being to prove two complementary concepts:

1. (REP) Establish a kind of "weak equivalent fibrant replacement" R : ∞-Cat.$\alpha \longrightarrow \infty$-Cat.$\alpha$ (.α gives the object component in Mathlib's category theory library), analogous to CW-complex replacement in Whitehead's original paper. It's especially nice if R forms the object component of a functor F : ∞-Cat $\longrightarrow$ ∞-Cat. D(F) : D(∞-Cat) $\longrightarrow$ D(∞-Cat) should be a categorical equivalence, and that is what we will do.

2. (HEP) For the object R X, demonstrate that any F,G : (R X) $\longrightarrow$ Y such that $\forall$(n:Nat),($\vec{\pi}_n$ F = $\vec{\pi}_n$ G), there is a directed homotopy equivalence between F and G. Note that "directed homotopy equivalence" consists of a composible sequence of simple directed homotopies H□ : $\Delta^1 \times$ (R X) $\longrightarrow$ Y, 1 □ i □ n, with even H□ running reverse to the odd H□.

Both of these will use induction on Lean's `Nat`. The first of these could be called a REP (for REplacement Property, but this isn't usual terminology), and the second typically uses induction and a HEP (Homotopy Extension Property). Our REPa will consist of objects made out of particular kinds of pushouts called attaching maps, and can be made functorial. Proving the HEPa can be done by well-order induction on the attaching maps present in our choice of R, thereby reducing to the case of extending a homotopy along a single attachment.

Our HEPa (directed box filling) is similar to the HEP shown in Whitehead's original paper, and to the approach detailed in Hatcher's textbook, though no doubt modified to suit our two goals:

(I) The analogue of the Puppe sequence on the front cover needs to hold.

(II) The first Whitehead theorem on the front cover needs to hold.

These two considerations determine our choice of $\vec{\pi}_n$, $\vec{\Omega}$, and $\vec{\omega}$. We take $\vec{\Omega}$ to be (simply) the internal hom functor $[\Delta^1$,-] (which requires showing that $\vec{\Omega}$X has the inner-horn filling condition). $\vec{\omega}$ is then defined as a certain pullback of $\vec{\Omega}$, and $\vec{\pi}_n$ is designed to produce a Puppe sequence with a meaningful notion of exactness by which we can demonstrate the goal of fixed point principals (i) and (ii). Specifically, it makes sense to use cubes in our definition of $\vec{\pi}_n$ because of how they are representing objects of $\vec{\Omega}^{n}$. Meanwhile, it is also clear that the quotient producing $\vec{\pi}_n$ is subtle in exactly how it requires fixing the endpoints of a sequence of alternating directed homotopies. We will define $\vec{\pi}_n$'s by identifying those objects x, y: $\vec{\Omega}^{n}$ X which are homotopic by a homotopy which restricts to a constant along the face maps f⍰ : $\vec{\Omega}^{n-1}$ X $\longrightarrow \vec{\Omega}^{n-1}$ X (which correspond to pairs (n,b), where b : Bool).

Imagine for a moment the picture of a square shaped cusion; we might make such a cusion by first soeing together 6 squares of cloth and filling it with material, then "soeing the walls down to a square". Here we go with this:

1. Define a n-cubical cusion using the boundary of an n-1 cube times Δ^1, i.e. the quotient of $(\Delta^1)^{n-1} \times \Delta^1$ by an equivalence relation, but we have to start our model somewhere), or perhaps more easily the pushout of f : $\Delta^1 \times$ (⍰$((\Delta^1)^{n})$) $\longrightarrow (\Delta^1)^{n+1}$ by the projection map $\Delta^1 \times$ (⍰$((\Delta^1)^{n})$) $\longrightarrow$ ⍰$((\Delta^1)^{n})$

2. Define a simplicial cusion using the boundary of an n-1 simplex times Δ^1, i.e. the quotient of (Δ^1) by an equivalence relation, or perhaps more easily the pushout of f : $\Delta^1 \times$ (⍰(Δ^{n})) $\longrightarrow (\Delta^1) \times \Delta^{n}$ by the projection map $\Delta^1 \times$ (⍰$((\Delta^1)^{n})$) $\longrightarrow$ ⍰(Δ^{n})

The boundary of a cusion is a pouch, isomorphic to a pushout of two cubes glued together at their boundaries:

1. Define a n-cubical pouch as the pushout of two boundary maps ⍰$((\Delta^1)^{n}) \longrightarrow (\Delta^1)^{n}$

2. Define a simplicial pouch as the pushout of two boundary maps ⍰$(\Delta^{n}) \longrightarrow \Delta^{n}$

Notice that paths in $\vec{\Omega}^{\mathrm{n}}$X produce paths in $\vec{\Omega}^{n-1}$X in as many ways as there are face maps $(\Delta^1)^{n-1} \longrightarrow \Delta^{1\,n}$, these could be called restrictions and are no doubt related to the pouches and cusions we just defined. The cartesian closed structure on simplicial sets with the lifting condition clarifies the relationship between the two available definitions of $\vec{\pi}_n$:

1. Homotopies of maps from a cube which are constant on the boundary
2. Paths of maps in $\vec{\Omega}^{n-1}$X which produce constant maps under the mentioned restritions.
3. Maps from a pouch mod an equivalence relation (really we phrase this as a pushout!), namely the equivalence relation in which any two maps from a pouch that extend to maps from a cusion are identified.

After we construct $\vec{\pi}_n$ in the first section, we will be in a place to demonstrate that the natural transformation `weak_equivalence : repl ⟶ (𝟙 ∞-Cat)` consists of weak equivalences (a fact which we call REP, which is short for REplacement Principal). This is covered in the section titled REP, which also constructs `repl` and `weak_requivalence`.

In sum, the goal of the present chapter is to use similar insights to the proof of the Whitehead theorem featured Hatcher's textbook to prove Wa and Pa for the model of quasicategories, using Mathlib's predefined horns and simplices in its simplicial sets section. The main difference is that our work must take care to respect the directed nature of quasicategories.

1. Defining `repl`
2.

14. REP

We have divided the work of proving Whitehead theorem (a) into two steps: REP and HEP. In this section, we construct a functor `repl : ∞-Cat ⟶ ∞-Cat` along with a natural transformation `weak_equivalence : repl ⟶ (𝟙 ∞-Cat)`. To construct `repl`

15. HEP

Consider the context of , supposing that we have constructed a homotopy ... This gives a picture that is a bit like “filling up a jar“: a homotopy h : of f, g : ⍰$\Delta^2 \longrightarrow$ Y, along with the value of g on Δ^2, produces a “jar” shape in Y, which can be “filled up” to produce a homotopy h : $\Delta^1 \times \Delta^2 \longrightarrow$ Y. This is easier for simplicial-based approaches than for point-set topological approaches, the latter of which needs extra steps that deform a map into a cellular map.

This construction, in the case of point set topology, often involves first deforming maps so as to be cellular; however our analogue of CW complexes allows us to skip this step.

This construction (HEP for quasicategories) may even be equivalent to the quasicategory lifting condition if we are lucky. It is also the main technical device allowing for our concrete choice of model (quasicategories).

In this section, we demonstrate this extension property and use it to conclude the Whitehead theorem for ∞-categories stated above.

Directed Prism Filling (DPF) Let Y be a quasicategory, and let f, g : ⍰$\Delta^n \longrightarrow$ Y. A homotopy h : ⍰$\Delta^n \times \Delta^1 \longrightarrow$ Y between f, g : ⍰$\Delta^n \longrightarrow$ Y extends to a map H : $\Delta^n \times \Delta^1$ $\longrightarrow$ Y; this follows from the condition that Y be a quasicategory. H(-,1) and g match on ⍰Δ^n, producing a map f : X $\longrightarrow$ Y, where X consists of two copies of Δ^n glued together at the boundary. Consider a space X’ formed as a quotient of $\Delta^n \times \Delta^1$ by ⍰Δ^n $\times \Delta^1$. There is a map ϕ : X $\longrightarrow$ X’. An induction hypothesis on f and g involving π_n ensures that the aparent map X $\longrightarrow$ Y lifts along ϕ, producing a map from $\Delta^n \times \Delta^1$ which is constant on ⍰$\Delta^n \times \Delta^1$. Stacking this on top of H can be done using an isomorphism between Δ^1 and Δ^1 glued with itself along different endpoints. Altogether this produces a homotopy between f and g.

Directed prism filling may combine fruitfully with the yoneda lemma and/or the fact that simplicial sets are determined by the sets [Δ^n,X] along with combinatorial information (face and degeneracy maps).

Decomposing $\Delta^n \times \Delta^1$ into a colimit involving n+1 Δ^{n+1}’s ...

In the above, it may be easier if we make use of sub-simplicial sets and prove the theorem using that colimit applied to a natural isomorphism of diagrams products an isomorphism.

The decomposition

A definition of $\vec{\pi}_n$ which is consistent with our goals of Wa and Pa is one as a certain pushout involving ($\vec{\Omega}^{\mathrm{n}}$ X)– one which amounts to taking an equivalence relation by paths in $\vec{\Omega}^{\mathrm{n}}$ X which restrict to constant paths along the face maps f⍰ : $\vec{\Omega}^{n-1}$ X $\longrightarrow$ $\vec{\Omega}^{\mathrm{n}}$ X. Here, $\vec{\Omega}$ is easy to define in the model of quasi-categories, and it amounts . Besides fullfilling our goal of the first Whitehead theorem and puppe sequence, this definition of $\vec{\pi}_n$ strikes me as elegant because it uses all of the ways for $\vec{\Omega}^{\mathrm{n}}$ X to map into $\vec{\Omega}^{n+1}$ X.

The next symbols in the project's "periodic table" that we construct, after $\vec{\Omega}$ and $\vec{\pi}_n$, will be $\vec{\mathrm{B}}$ and $\vec{\mathrm{E}}$, which we feature in the chapter on Puppe sequence (a).

A useful thing for us to construct first is the boundary of a product of Δ^1's and the boundary of a directed simplex. We might even like to expand on this later, but for now just consider for a moment how each might be made out of a glueing construction involving face maps.

Even though the $\vec{\pi}_n$'s can be defined using $\vec{\Omega}^{\mathrm{n}}$ X and various face maps f_(n, b) : $\vec{\Omega}^{n-1}$ X $\longrightarrow$ $\vec{\Omega}^{n}$ X for $b : \{0, 1\}$, it may be nice to have this as a result, with the definition one featuring two cubes glued together along their boundary.

This means that we want directed box filling in addition to directed prism filling (but which also uses directed prism filling in its proof).

Directed Box Filling (DBF) Let Y be a quasicategory, and let f, g : ⍰Δ^{n} $\longrightarrow$ Y. A homotopy h : ⍰$\Delta^{\mathrm{n}} \times \Delta^1$ $\longrightarrow$ Y between f, g : ⍰Δ^{n} $\longrightarrow$ Y extends to a map H : $\Delta^{\mathrm{n}} \times \Delta^1$ $\longrightarrow$ Y; this follows from the condition that Y be a quasicategory. H(-,1) and g match on ⍰Δ^{n}, producing a map f : X $\longrightarrow$ Y, where X consists of two copies of Δ^{n} glued together at the boundary. Consider a space X' formed as a quotient of $\Delta^{\mathrm{n}} \times \Delta^1$ by ⍰Δ^{n} $\times \Delta^1$. There is a map ϕ : X $\longrightarrow$ X'. An induction hypothesis on f and g involving π_n ensures that the aparent map X $\longrightarrow$ Y lifts along ϕ, producing a map from $\Delta^{\mathrm{n}} \times \Delta^1$ which is constant on ⍰$\Delta^{\mathrm{n}} \times \Delta^1$. Stacking this on top of H can be done using an isomorphism between Δ^1 and Δ^1 glued with itself along different endpoints. Altogether this produces a homotopy between f and g.

This goes hand-in-hand with a definition of $\vec{\pi}_n$ which suits (I) and (II) in the introduction to chapter (3). If we make sure to prove lemmas...

The box filling and prism filling HEPs can be extended to the case of attaching all cells of a particular fixed dimension and as indexed by simplicial set arising from a set (or Lean 4 `Type`). That is, we might like to extend $\times$ () (or possibly somehow a `Set` as well), and that we may find an interest in the following two definitions of $\vec{\pi}_n$, which are designed to fullfill both (I) and (II) in the chapter's introduction.

Breaking down DBF further can be done conveniently using sub-simplicial sets, just like we used in the proof of prism filling.

Decomposing $(\Delta^1)^n$ into a colimit involving n! Δ^n's Consider the face maps f� : $\Delta^n \longrightarrow \Delta^{n+1}$

The decomposition The box filling lemma allows us to prove HEP:

16. The Whitehead Theorem for ∞-Cat

The HEP in the last

..H(-,1) and g match on $\partial\Delta^{n}$, producing a map $f : X \longrightarrow Y$, where X consists of two copies of Δ^{n} glued together at the boundary. Consider a space X' formed as a quotient of $\Delta^{n} \times \Delta^{1}$ by $\partial\Delta^{n} \times \Delta^{1}$. There is a map $\phi : X \longrightarrow X'$. An induction hypothesis on f and g involving π_n ensures that the aparent map $X \longrightarrow Y$ lifts along ϕ, producing a map from $\Delta^{n} \times \Delta^{1}$ which is constant on $\partial\Delta^{n} \times \Delta^{1}$. Stacking this on top of H can be done using an isomorphism between Δ^{1} and Δ^{1} glued with itself along different endpoints. Altogether this produces a homotopy between f and g.

Imagine

Chapter 4: Internal categories and internal presheaves

In this chapter, we discuss internal categories and internal presheaves in a pullback system. We may keep in mind that internal categories and internal presheaves can be formed in any category with pullbacks, even though we focus on the case of pullback systems because of our interest in Whitehead theorem (a).

After defining the category of internal categories D(Γ), we proceed to observe how, for C , D : D(Γ), F : C $\longrightarrow$ D, ($\vec{\omega}$ F).obj F forms an internal categry. Further, in considering internal ($\vec{\mathrm{P}}$_(Γ) F)-presheaves for C , D : D(Γ), F : C $\longrightarrow$ D, we proceed to make observations about ($\vec{\omega}$ F).obj G.

Section	Description
`IntCat Γ : Cat`	Internal categories
`IntShf Γ C : Cat`	Internal C-presheaves
The internal category principal	`f ×_(B) f` forms an internal category
The internal presheaf principal	`f ×_(B) f` forms an internal presheaf
$\vec{\mathrm{P}}$ C : IntCat D(∞-Cat)	$\vec{\Omega}$ C forms a component of an internal category
$\vec{\mathrm{p}}$ ($\mathbb{1}$ C) D : IntShf D(∞-Cat⁄C) ($\vec{\mathrm{P}}$ C)	$\vec{\omega}$ ($\mathbb{1}$ C) D forms a component of an internal C-presheaf

17. IntCat Γ

In this chapter we define an internal category. Internal categories are most commonly defined on categories with enough pullbacks, but here we may also like to keep in mind that it is valuable to be able to iterate `IntCat` in the way of composition of functors, ideally without having to handle any technicalities as to its being well defined. In other words, we define a the category of internal categories for all categories, with the internal category construction thereby producing an endofunctor `IntCat` of Mathlib's category of categories. Hence we add to the information of the structure an assumed existence of the various pullbacks involved. This is not any sort of issue, since we only name the internal categories for which the relevant pullback does exist.

Lean 25

```
-- definition of an internal category in a pullback
↪  system
/-
structure internal_category (Γ : Cat) where
  Obj : .Obj
  Mor : .Obj
  Dom : .Hom Mor Obj
  Cod : .Hom Mor Obj
  Idn : .Hom Obj Mor
  Fst : .Cmp Obj Mor Obj Idn Dom  = 𝟙_(Γ.Obj) Obj
  Snd : .Cmp Obj Mor Obj Idn Cod = 𝟙_(Γ.Obj) Obj
--  Cmp : D(Γ).Γ.PulObj ...
-- Id₁ : D(Γ).
-- Id₂ : D(Γ).
-- Ass : D(Γ).
-/
```

The internal functor structure combines with the internal category structure to give a category of internal categories in a pullback system.

Lean 26

```
-- definition of an internal functor in a pullback
↪  system
structure internal_functor (Γ : pullback_system) (C :
↪  internal_category Γ) (D : internal_category Γ)
↪  where
  Obj : D(Γ).Hom C.Obj D.Obj
-- Mor : D(Γ).
-- Fst : D(Γ).
-- Snd : D(Γ).
-- Idn : D(Γ).
-- Cmp : D(Γ).
```

Lean 27

```
-- definition of the identity internal functor in a
↪  pullback system
def IntCatIdn (Γ : pullback_system) (C :
↪  internal_category Γ) : (internal_functor Γ C C)
↪  := sorry
```

Lean 28

```
-- definition of the composition of internal
↪  functors in a pullback system
def IntCatCmp (Γ : pullback_system) (C :
↪  internal_category Γ) (D : internal_category Γ) (E
↪  : internal_category Γ) (F : internal_functor Γ C
↪  D) (G : internal_functor Γ D E) :
↪  (internal_functor Γ C E) := sorry
```

Lean 29

```
-- proving the the first identity law for internal
↪  categories in a pullback system
def IntCatId₁ (Γ : pullback_system) (X :
↪  internal_category Γ) (Y : internal_category Γ) (f
↪  : internal_functor Γ X Y) : IntCatCmp Γ X Y Y f
↪  (IntCatIdn Γ Y) = f := sorry
```

Lean 30

```
-- proving the second identity law for internal
↪ categories in a pullback system
def IntCatId₂ (Γ : pullback_system) (X :
↪ internal_category Γ) (Y : internal_category Γ) (f
↪ : internal_functor Γ X Y) : (IntCatCmp Γ X X Y
↪ (IntCatIdn Γ X) f = f) := sorry
```

Lean 31

```
-- proving the associativity law for internal
↪ categories in a pullback system
def IntCatAss (Γ : pullback_system) (W :
↪ internal_category Γ) (X : internal_category Γ) (Y
↪ : internal_category Γ) (Z  : internal_category Γ)
↪ (f : internal_functor Γ W X) (g :
↪ internal_functor Γ X Y) (h : internal_functor Γ Y
↪ Z) : IntCatCmp Γ W X Z f (IntCatCmp Γ X Y Z g h)
↪ = IntCatCmp Γ W Y Z (IntCatCmp Γ W X Y f g) h :=
↪ sorry
```

Lean 32

```
/-
def IntCat (Γ : pulback_system) : Cat.Obj := {Obj
↪ := internal_category Γ, Hom :=
↪ internal_functor Γ, Idn := IntCatIdn Γ, Cmp :=
↪ IntCatCmp Γ, Id₁ := IntCatId₁ Γ, Id₂ :=
↪ IntCatId₂ Γ, Ass := IntCatAss Γ}
-/
```

Lean 33

```
-- notation : 2000 "Cat_(" Γ ")" => IntCat Γ
```

18. IntShf Γ C

The mentioned book `Galois Theories` by Janelidze and Borceux features a definition of internal presheaves for an internal groupoid in chapter 7 which makes a good reference for the present discussion.

Lean 34

```
-- internal C-presheaves
def internal_presheaf (Γ : pullback_system) (C :
↪ (IntCat Γ).Obj) : Type := sorry
```

Lean 35

```
-- defining an internal functor between internal
↪ C-presheaves
def ShfHom (Γ : pullback_system) (C : (IntCat Γ).Obj)
↪ (F : internal_presheaf Γ C) (G :
↪ internal_presheaf Γ C) : Type := sorry
```

Lean 36

```
-- defining the identity internal functor of an
↪ internal C-sheaf
def ShfIdn (Γ : pullback_system) (C : (IntCat Γ).Obj)
↪ (F : internal_presheaf Γ C) : ShfHom Γ C F F :=
↪ sorry
```

Lean 37

```
-- defining the composition of internal functors
def ShfCmp (Γ : pullback_system) (C : (IntCat Γ).Obj)
↪ (F : internal_presheaf Γ C) (G :
↪ internal_presheaf Γ C) (H : internal_presheaf Γ
↪ C) (f : ShfHom Γ C F G)  (g : ShfHom Γ C G H) :
↪ ShfHom Γ C F H := sorry
```

Lean 38

```
-- proving the first identity law for internal
↪  functors
def ShfId₁ (Γ : pullback_system) (C : (IntCat Γ).Obj)
↪  (X : internal_presheaf Γ C) (Y :
↪  internal_presheaf Γ C) (f : ShfHom Γ C X Y) :
↪  ((ShfCmp Γ C X Y Y f (ShfIdn Γ C Y)) = f) :=
↪  sorry
```

Lean 39

```
-- proving the second identity law for internal
↪  functors
def ShfId₂ (Γ : pullback_system) (C : (IntCat Γ).Obj)
↪  (X : internal_presheaf Γ C) (Y :
↪  internal_presheaf Γ C) (f : ShfHom Γ C X Y) :
↪  ((ShfCmp Γ C X X Y (ShfIdn Γ C X) f) = f) :=
↪  sorry
```

Lean 40

```
-- proving the associativity law for internal
↪  functors
def ShfAss (Γ : pullback_system) (C : (IntCat Γ).Obj)
↪  (W : internal_presheaf Γ C) (X :
↪  internal_presheaf Γ C) (Y : internal_presheaf Γ
↪  C) (Z : internal_presheaf Γ C) (f : ShfHom Γ C W
↪  X) (g : ShfHom Γ C X Y) (h : ShfHom Γ C Y Z) :
↪  (ShfCmp Γ C) W X Z f ((ShfCmp Γ C) X Y Z g h) =
↪  (ShfCmp Γ C) W Y Z ((ShfCmp Γ C) W X Y f g) h :=
↪  sorry
```

Lean 41

```
def IntShf (Γ : pullback_system) (C : (IntCat Γ).Obj)
↪  : Cat.Obj := {Obj := internal_presheaf Γ C, Hom
↪  := ShfHom Γ C, Idn := ShfIdn Γ C, Cmp := ShfCmp Γ
↪  C, Id₁ := ShfId₁ Γ C, Id₂ := ShfId₂ Γ C, Ass :=
↪  ShfAss Γ C}
```

Lean 42

```
notation : 2000 "Shf_(" Γ ")" => IntShf Γ
```

Next we approach the internal category principal and internal presheaf principals, which concern how (directed) homotopy pullback can produce internal categories and internal presheaves.

19. The Internal Category Principal

In this section we mention the internal category principal, which says that the pullback of any morphism with itself forms a component of an internal category in any category with pullbacks. In fact, the most general form of the theorem works for a noncommutative analogue of pullback, wheras the case of pullback gives an internal groupoid (a fact which we will show in later chapters).

20. The Internal Presheaf Principal

Next we mention the internal presheaf principal, which says that the pullback of any morphism with another forms a component of an internal presheaf in any category with pullbacks. Just as is the case for the last theorem, the most general form of this idea works for non-commutative analogues of pullback, wheras the case of pullback gives an internal groupoid action.

21. P

In this section, we construct the functor $\vec{\mathrm{P}}$ mentioned in the introduction. Specifically, `(Ω f)` forms a component of an internal category.

Later we will add a theorem to the effect that $\vec{\mathrm{P}}$ as constructed is naturally isomorphic to a functor constructed using Eilenberg-Moore operations (speficially the structureΩ map of the Eilenberg-Moore category of a monad corresponding to $\vec{\Omega}$).

Lean 43

```
-- def path_spaceObj (Γ : pullback_system) (E :
↪  Γ.Obj.Obj) (B : Γ.Obj.Obj) (f : Γ.Obj.Hom E B) :
```

Lean 44

```
-- def path_spaceHom (Γ : pullback_system) (E :
↪  Γ.Obj.Obj) (B : Γ.Obj.Obj) (f : Γ.Obj.Hom E B) :
```

Lean 45

```
-- def path_spaceDom (Γ : pullback_system) (E :
↪  Γ.Obj.Obj) (B : Γ.Obj.Obj) (f : Γ.Obj.Hom E B) :
```

Lean 46

```
-- def path_spaceCod (Γ : pullback_system) (E :
↪  Γ.Obj.Obj) (B : Γ.Obj.Obj) (f : Γ.Obj.Hom E B) :
```

Lean 47

```
-- def path_spaceIdn (Γ : pullback_system) (E :
↪  Γ.Obj.Obj) (B : Γ.Obj.Obj) (f : Γ.Obj.Hom E B) :
```

Lean 48

```
-- def path_spaceFst (Γ : pullback_system) (E :
↪  Γ.Obj.Obj) (B : Γ.Obj.Obj) (f : Γ.Obj.Hom E B) :
```

Lean 49

```
-- def path_spaceSnd (Γ : pullback_system) (E :
↪  Γ.Obj.Obj) (B : Γ.Obj.Obj) (f : Γ.Obj.Hom E B) :
```

Lean 50

```
-- def path_spaceCmp (Γ : pullback_system) (E :
↪  Γ.Obj.Obj) (B : Γ.Obj.Obj) (f : Γ.Obj.Hom E B) :
```

Lean 51

```
-- def path_spaceId₁ (Γ : pullback_system) (E :
↪  Γ.Obj.Obj) (B : Γ.Obj.Obj) (f : Γ.Obj.Hom E B) :
```

Lean 52

```
-- def path_spaceId₂ (Γ : pullback_system) (E :
↪  Γ.Obj.Obj) (B : Γ.Obj.Obj) (f : Γ.Obj.Hom E B) :
```

Lean 53

```
-- def path_spaceAss (Γ : pullback_system) (E :
↪  Γ.Obj.Obj) (B : Γ.Obj.Obj) (f : Γ.Obj.Hom E B) :
↪  := sorry
```

Lean 54

```
def path_space (Γ : pullback_system) (E : Γ.Obj.Obj)
↪  (B : Γ.Obj.Obj) (f : Γ.Obj.Hom E B) : (IntCat
↪  Γ).Obj := sorry
/-
{Obj := path_spaceObj, Hom := path_spaceHom, Idn
↪  := path_spaceIdn, Cmp := path_spaceCmp, Id₁ :=
↪  path_spaceId₁, Id₂ := path_spaceId₂, Ass :=
↪  path_spaceAss}
-/
```

Lean 55

```
notation "P_(" Γ ")" => path_space Γ
```

22. p

In this final section of the chapter, we establish the internal presheaf principal, which says that `(ω f).obj g` forms a component of an internal `P f`-presheaf $\vec{\omega}$ (which produces an internal presheaf). We write `ω_(Γ) f g : Shf_(Γ) (P_(Γ) f)` for this internal presheaf.

The descent principal expresses how

Lean 56

```
-- assembling the descent equivalence
/-
def descent_principal (Γ : pullback_system) (E :
↪  Γ.Obj.Obj) (B : Γ.Obj.Obj) (f : Γ.Obj.Hom E
↪  B) : Type :=  (!_(Cat) (?_(Cat) ( (M E B
↪  f))))).Cod ≃_(Cat) (IntShf Γ) (P_(Γ) E B f)
-/
```

Chapter 5: The Puppe Sequence for ∞-Categories

In this chapter we construct the Puppe sequence for $\vec{\pi}_n$. Note: one joint in this exact sequence consists not of a map but an action.} This will be used in the next chapter two establish two of the six categorical equivalences.

Chapter 6: The Categorical Fixed Point Principals

After the construction in chapter 11, we will prove the internal category fixed point principal, which is the first categorical equivalence of the six mentioned in the introduction. We also prove in this chapter the internal C-presheaf fixed point principal, which is the second categorical equivalence of the six mentioned in the introduction. To do this, we first define $\vec{B} = \vec{B}$_(∞-$\mathbb{C}\mathbb{a}\mathbb{t}$) and $\vec{b} = \vec{b}$_(∞-$\mathbb{C}\mathbb{a}\mathbb{t}$).

This much may be possible for the case of simplicial sets using first the construction of $\vec{E}$ as a directed homotopy colimit (we can use Mathlib's geometric realization), and then quotienting by an apparant action of a particular internal category.

23. B

Lean 57

```
-- def B : (Cat.Hom Cat_(∞-ℂ𝕒𝕥) D(∞-ℂ𝕒𝕥)).Obj :=
↪  sorry

--
```

24. b

The `b` symbol formally gives a pseudofunctor, but we can also create a model in which it is a functor. It occurs as one side of a categorical equivalence, the second of the six categorical equivalences which I have called "fixed point principals".

Lean 58

```
-- def Par (C : D(∞-𝕮𝖆𝖙).Obj) : Shf_(∞-𝕮𝖆𝖙)
↪  (P_(∞-𝕮𝖆𝖙) C C (𝟙_(D(∞-𝕮𝖆𝖙)) C)) ⟶
↪  (Cmp_(∞-𝕮𝖆𝖙) C) := sorry
```

Lean 59

```
-- notation "b" => Par
```

25. The Internal Category Fixed Point Principal

The internal category fixed point principal will look something like this:

Lean 60

```
-- def internal_category_fixed_point_principal :
↪  Type := D(∞-ℂ𝕒𝕥) ≃ (IntCat D(∞-ℂ𝕒𝕥))
```

It should be readily available from the construction in the last chapter.

Lean 61

```
-- def internal_category_fixed_point_principal_proof
↪  : internal_category_fixed_point_principal :=
↪  {Fst :=
↪  internal_category_fixed_point_principalFst, Snd
↪  := internal_category_fixed_point_principalSnd,
↪  Id₁ :=
↪  internal_category_fixed_point_principalId₁, Id₂
↪  := internal_category_fixed_point_principalId₂}
```

26. The Internal Presheaf Fixed Point Principal

The internal presheaf fixed point principal consists of a categorical equivalence between D(∞-Cat/C) and internal C-presheaves in D(∞-Cat/C).

Lean 62

```
def internal_presheaf_fixed_point_principal (C :
↪  D(∞-ℂ𝕒𝕥).Obj) : Type := Shf_(∞-ℂ𝕒𝕥)
↪  (P_(∞-ℂ𝕒𝕥) C C (𝟙_(D(∞-ℂ𝕒𝕥)) C)) ≃_(Cat)
↪  (!_(Cat) (?_(Cat) (¡_(Cat) (¿_(Cat) (p_(∞-ℂ𝕒𝕥) C
↪  C (𝟙_(D(∞-ℂ𝕒𝕥)) C)))))).Cod
```

Next we prove the internal C-sheaf fixed point principal. This says says that `Shf_(∞-ℂ𝕒𝕥) (P_(∞-ℂ𝕒𝕥) C C (𝟙_(D(∞-ℂ𝕒𝕥)) C)) ≃_(Cat) !? (p_(∞-ℂ𝕒𝕥) C C (𝟙_(D(∞-ℂ𝕒𝕥)) C))`.

Lean 63

```
-- The internal C-sheaf fixed point principal

def internal_presheaf_fixed_point_principal_proof (C
↪  : D(∞-ℂ𝕒𝕥).Obj) :
↪  internal_presheaf_fixed_point_principal C :=
↪  {Fst :=
↪  internal_presheaf_fixed_point_principalFst, Snd
↪  := internal_presheaf_fixed_point_principalSnd,
↪  Id₁ :=
↪  internal_presheaf_fixed_point_principalId₁, Id₂
↪  := internal_presheaf_fixed_point_principalId₂}
```

PART 2: ∞-GROUPOIDS

Chapter 7: ∞-Grpd

In this section we establish the categories D(∞-Grpd) and D(∞-Grpd/G) for G : D(∞-Grpd) out of the previous constructions. Our model for these categories is directly based on Mathlib's category of simplicial sets with the Kan lifting condition.

Lean 64

```
--def derived_category_of_infinity_groupoids : Cat
↪  := sorry
```

Lean 65

```
/-
notation for D(∞-Grpd)
-/
```

Lean 66

```
--def derived_category_of_infinity_groupoids_over (G
↪  : D(∞-𝔾𝕣𝕡𝕕)) : Cat := sorry
```

Lean 67

```
/-
notation for D(∞-Grpd/G)
-/
```

Chapter 8: The Whitehead Theorem for ∞-Groupoids

In this section, we prove Whitehead theorem (b), which says that $\forall$(E:D(∞-Grpd)),$\forall$(B:D(∞-Grpd)),$\forall$(F:E $\longrightarrow$ B),$\forall$(G:E $\longrightarrow$ B),($\forall$(n:Nat),($\overleftrightarrow{\pi}_n$ F = $\overleftrightarrow{\pi}_n$ G)) $\longrightarrow$ F = G, where $\overleftrightarrow{\pi}_n$ is notation for $\overleftrightarrow{\pi}$ n.

The main idea here is to treat this by induction, extending a homotopy for each n to a homotopy for n+1. This gives a picture that is a bit like "filling up a jar": a homotopy h : I $\times$ bΔ^2 of f, g : bΔ^2 $\longrightarrow$ Y, along with the value of g on Δ^2, produces a "jar" shape in Y, which can be "filled up" to produce a homotopy h : I $\times$ Δ^2 $\longrightarrow$ Y. This is easier for simplicial-based approaches than for point-set topological approaches, the latter of which needs extra steps that deform a map into a cellular map.

Alternatively, we can probably use the homotopy extension property shown for quasicategories in the first place, thereby recycling old work.

27. HEP for ∞-groupoids

In this section, we demonstrate this extension property and use it to conclude the Whitehead theorem for ∞-categories stated above, using the previously constructed simplicial set based model of D(∞-Cat) and D(∞-Cat/C).

28. The Whitehead theorem for ∞-groupoids

The “jar filling” lemma of the last section can be applied to our analogue of CW-complexes (simplicial sets formed by gluing simplices Δ^n along their boundaries). Potentially we will use a well order somehow reducing to the case of homotopy-extension to a single jar.

29. Ω

$\overleftrightarrow{\Omega}$, the analogue of loop space, is the internal hom functor [I,-] : ∞-Grpd $\longrightarrow$ ∞-Grpd.

30. ω

$\overleftrightarrow{\Omega}$ is to internal groupoids as $\overleftrightarrow{\omega}$ is to internal G-presheaves. $\overleftrightarrow{\omega}$ is also called homotopy pullback, but this by no means standard notation for homotopy pullback at all. These functors will later be used to produce functors $\vec{\mathrm{P}} : \mathrm{D}(\infty\text{-Cat}) \longrightarrow \mathrm{IntCat}\ \mathrm{D}(\infty\text{-Cat})$ and $\vec{\mathrm{p}} : \mathrm{D}(\infty\text{-Cat}/\mathrm{C}) \longrightarrow \mathrm{IntShf}\ \mathrm{D}(\infty\text{-Cat}/\mathrm{C})\ (\vec{\mathrm{P}}\ \mathrm{C})$.

31. π_0

Chapter 9: Internal Groupoids and their Actions

In this next section, we continue the approach to Whitehead theorem (b) by defining the category of internal groupoids and internal G-presheaves.

32. IntGrpd Γ

Internal groupoids can be defined in any category with pullbacks.

Lean 68

```
structure internal_groupoid (Γ : category with
 ↪  pullbacks) where
  Obj := D(Γ).Obj
-- Dom :=
-- Cod
-- Idn
-- Fst
-- Snd
-- Cmp
-- Id₁
-- Id₂
-- Ass
-- Com
```

Lean 69

Lean 70

Lean 71

Lean 72

Lean 73

Lean 74

Lean 75

```
def IntGrpd (Γ : pulback_system) : Cat.Obj := sorry
```

Lean 76

```
notation "Grpd_(" Γ  ")" => IntGrpd Γ
```

33. IntAct Γ G

Here we define internal groupoid actions.

Lean 77

```
structure groupoid_presheaf (Γ : category with
↪ pullbacks) (G : internal_groupoid Γ) where
  Obj : D(Γ).Obj
-- Mor : D(Γ).
-- Dom : D(Γ).
-- Cod : D(Γ).
-- Fst : D(Γ).
-- Snd : D(Γ).
-- Idn : D(Γ).
-- Idn : D(Γ).
-- Cmp : D(Γ).
-- Id₁ : D(Γ).
-- Id₂ : D(Γ).
-- Ass : D(Γ).
-- Com : D(Γ).
```

Lean 78

```
def ActHom (Γ : pullback_system) (X :
↪ groupoid_action Γ)
```

Lean 79

Lean 80

Lean 81

Lean 82

Lean 83

34. The Internal Groupoid Principal

The internal groupoid principal stems from the simple observation that the pullback of a map by itself (minding matters of existence of pullback for a moment) forms the morphism component of an internal groupoid. It already been observed that it forms the morphism component of an internal category. Here, we also extend the observation that the derived homotopy pullback of an ∞-functor between ∞-groupoids by itself forms, in a derived category, an internal groupoid.

35. The Internal Groupoid Action Principal

The internal groupoid action principal stems from the simple observation that the pullback of a map by another forms the morphism component of an internal groupoid action. It already been observed that it forms the morphism component of an internal C-presheaf. Here, we also extend the observation that the derived homotopy pullback of an ∞-functor between ∞-groupoids by another forms, in a derived category, an internal groupoid action.

36. P

This section will construct $\overleftrightarrow{\mathrm{P}}$, which is an internal groupoid that one obtains from any ∞-groupoid.

37. p

This section will construct the functor $\vec{\mathrm{p}}$ mentioned in the introduction. Later we will add a theorem stating that this functor is infact naturally isomorphic to a functor constructed using $\overleftrightarrow{\omega}$ and constructiosn from Eilenberg-Moore theory.

Chapter 10: The Puppe Sequence for ∞-Groupoids

In this chapter we construct the Puppe sequence for $\vec{\pi}_n$. Note: one joint in this exact sequence consists not of a map but an action.} This will be used in the next chapter two establish the second of the six categorical equivalences mentioned in the introduction.

Chapter 11: The Groupoid Fixed Point Principals

38. B

After the last section is complete, we will be in a place to prove the internal groupoid and internal groupoid presheaf fixed point principals, which are the third and fourth of the six categorical equivalences mentioned in the introduction.

Lean 84

```
-- def BInfGrpd : (Cat.Hom Grpd_(∞-𝔾𝕣𝕡𝕕)
↪  D(∞-𝔾𝕣𝕡𝕕)).Obj := sorry
```

39. b

Lean 85

```
-- def Par (C : D(∞-ℂ𝕒𝕥).Obj) : Shf_(∞-ℂ𝕒𝕥)
↪  (P_(∞-ℂ𝕒𝕥) C C (𝟙_(D(∞-ℂ𝕒𝕥)) C)) ⟶
↪  (Cmp_(∞-ℂ𝕒𝕥) C) := sorry
```

Lean 86

```
-- notation "b" => Par
```

40. The Internal Groupoid Fixed Point Principal

Lean 87

```
def internal_groupoid_fixed_point_principal (Γ :
↪  pulback_system) : Type := D(Γ) ≃_(Cat) Grpd_(Γ)
```

Lean 88

```
-- def internal_groupoid_fixed_point_principal_proof
↪  : internal_category_fixed_point_principal ∞-ℂ𝕒𝕥
↪  := sorry
```

41. The Internal Presheaf Fixed Point Principal

Lean 89

```
-- def
↪  internal_groupoid_presheaf_fixed_point_principal
↪  (Γ : pullback_system) (C : D(Γ).Obj) : Type :=
↪  Shf_(Γ) (P (𝟙_(D(Γ)) C)) ≅ Der_(Γ) C
```

PART 3: BASED CONNECTED ∞-GROUPOIDS

Chapter 12: ∞-Grpd$_0$

Here we define the mentioned categories $D(\infty\text{-Grpd}_0)$ of connected based ∞-groupoids and $D(\infty\text{-Grpd}_0/G_0)$ mentioned in the introduction.

42. Ω

43. ω

44. π_0

Chapter 13: The Whitehead Theorem

In this chapter we prove the following (which we have called Whitehead Theorem (c)): $\forall$(E:D(∞-Grpd$_0$)),$\forall$(B:D(∞-Grpd$_0$)),$\forall$(f:E $\longrightarrow$ B),$\forall$(G:E $\longrightarrow$ B),($\forall$(n:Nat),(π_n F = π_n G)) $\longrightarrow$ F = G, where π_n is notation for π n.

This can be shown using CW-replacement and induction on n. Fibrant replacement of an object X entails replacing an object in ∞-Grpd$_0$ with a CW-object (an object made by successively glueing in higher and higher simplices along their boundaries obtaining a sequence X$_n$). Given an equality $\pi_{n+1}(f)$ =$\pi_{n+1}(g)$ and a homotopy equivalence h$_n$: I $\times$ X$_n$ $\longrightarrow$ Y between $f|_{X_n}, g|_{X_n}$: X$_n$ $\longrightarrow$ Y, we construct an extension of the homotopy equivalence I $\times$ X_{n+1} $\longrightarrow$ Y.

45. HEP for based connected ∞-groupoids

This HEP should follow from the last one.

46. The Whitehead theorem

Here we show the Whitehead theorem proper.

Chapter 14: Internal Groups

47. Grp_(Γ)

Lean 90

```
/-
structure internal_group ... where
  Obj := D(Γ).Obj
-- Dom :=
-- Cod
-- Idn
-- Fst
-- Snd
-- Cmp
-- Id₁
-- Id₂
-- Ass
-- Com
-/
```

Lean 91

Lean 92

Lean 93

Lean 94

Lean 95

Lean 96

Lean 97

```
-- def IntGrp (Γ : pulback_system) : Cat.Obj :=
↪  sorry
```

Lean 98

```
-- notation "Grp_(" Γ  ")" => IntGrp Γ
```

48. Act_(Γ) G

Here we define internal group actions. These will be important when we talk about G-principal bundles (themselves defined as internal group actions in the derived category of an overcategory).

Lean 99

```
/-
structure group_action (Γ : pullback_system) (G :
 ↪  internal_groupoid Γ) where
  Obj : D(Γ).Obj
-- Mor : D(Γ).
-- ...
-/
```

Lean 100

```
/-
def ActHom (Γ : pullback_system) (X :
 ↪  groupoid_action Γ)
-/
```

Lean 101

```
```

Lean 102

```
```

Lean 103

```
```

Lean 104

Lean 105

49. The Internal Group Principal

The internal group principal stems from the simple observation that the loop space forms a component of an internal group.

50. The Internal Group Action Principal

The internal group actions principal stems from the simple observation that the homotopy fiber forms a component of an internal group action.

51. P

This section will construct P, which is an internal group that one obtains from any based connected ∞-groupoid.

52. p

This section will construct the functor p mentioned in the introduction. Later we will add a theorem stating that this functor is in fact naturally isomorphic to a functor constructed using ω and using constructions from Eilenberg-Moore theory.

Chapter 15: The Puppe Sequence for Based Connected ∞-Groupoids

This chapter establishes the well know Puppe sequence for the based homotopy groups π_n. This is the well known Puppe sequence of homotopy groups.

Chapter 16: The Group Fixed Point Principals

53. B

`B` is the ordinary classifying space, and it is defined on internal groups in D(∞-Grpd$_0$).

Lean 106

```
-- def BInfGrpd : (Cat.Hom Grpd_(∞-𝔾𝕣𝕡𝕕)
↪  D(∞-𝔾𝕣𝕡𝕕)).Obj := sorry
```

54. b

`B` is the ordinary classifying space, and it is defined on internal group actions in D(∞-Grpd$_0$).

Lean 107

```
-- def Par (C : D(∞-𝕮𝖆𝖙).Obj) : Shf_(∞-𝕮𝖆𝖙)
↪   (P_(∞-𝕮𝖆𝖙) C C (𝟙_(D(∞-𝕮𝖆𝖙)) C)) ⟶
↪   (Cmp_(∞-𝕮𝖆𝖙) C) := sorry
```

Lean 108

```
-- notation "b" => Par
```

55. The Internal Group Fixed Point Principal

For a based connected space X, the path space [I,X] is weak equivalent to the loop space ΩX. This observation will allow us to prove that the category of based connected ∞-groupoids is internal groups in itself.

Lean 109

```
def internal_groupoid_fixed_point_principal (Γ :
↪  pulback_system) : Type := D(Γ) ≃_(Cat) Grpd_(Γ)
```

Lean 110

Lean 111

Lean 112

Lean 113

Lean 114

Lean 115

Lean 116

Lean 117

Lean 118

Lean 119

```
-- def
↪  internal_category_fixed_point_principal_proofId₂
↪  :
```

Lean 120

```
-- def internal_category_fixed_point_principal_proof
↪  : internal_category_fixed_point_principal ∞-ℂ𝕒𝕥
↪  := sorry
```

56. The Internal Group Action Fixed Point Principal

For a based connected space X, a based connected space Y, and a based map f : X $\longrightarrow$ Y, the homotopy pullback of f with 𝟙 Y is weak equivalent the homotopy pullback with the base. This fascinating insight

Lean 121

```
def internal_groupoid_action_fixed_point_principal
↪  (Γ : pullback_system) (C : D(Γ).Obj) : Type :=
↪  Shf_(Γ) (P_(Γ) C C (𝟙_(D(Γ)) C)) ≃_(Cat) Der_(Γ)
↪  C
```

Lean 122

Lean 123

Lean 124

Lean 125

Lean 126

Lean 127

Lean 128

Lean 129

Lean 130

Lean 131

Lean 132

```
-- def
↪ internal_groupoid_action_fixed_point_principal_proof
↪ (C : D(∞-ℂ𝕒𝕥).Obj) :
↪ internal_presheaf_fixed_point_principal ∞-ℂ𝕒𝕥 C
↪ := sorry
```

Lean 133

Works Cited

1. Borceux, F., and Janelidze, G. Galois Theories. Cambridge Studies in Advanced Mathematics, vol. 72. Cambridge University Press, Cambridge, 2001. ISBN 0-521-80309-8.

2. Tom Leinster, Higher Operads, Higher Categories, London Mathematical Society Lecture Note Series, vol. 298, Cambridge University Press, 2004.

3. Lurie, Jacob. Higher Topos Theory. Annals of Mathematics Studies, vol. 170. Princeton University Press, Princeton, NJ, 2009.

4. Leonardo de Moura and Jeremy Avigad, "The Lean Theorem Prover," Journal of Formalized Reasoning, vol. 8, no. 1, pp. 1-37, 2015.

5. Leonardo de Moura and Soonho Kong, "Lean Theorem Proving Tutorial," Proceedings of the 6th International Conference on Interactive Theorem Proving (ITP), Lecture Notes in Computer Science, vol. 9236, pp. 378-395, Springer, Berlin, 2015.

6. Jeremy Avigad, Leonardo de Moura, and Soonho Kong, "Theorem Proving in Lean," Logical Methods in Computer Science, vol. 12, no. 4, pp. 1-43, 2016.

7. Daniel Selsam, Leonardo de Moura, David L. Dill, and David L. Vlah, "Leonardo: A Solver for MIP and Mixed Integer Nonlinear Programming," Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS), pp. 493-504, 2019.

Further reading:

1. J. Beck, "Distributive laws," in Seminar on Triples and Categorical Homology Theory, Springer-Verlag, 1969, pp. 119-140.

2. Saunders Mac Lane, "Categories for the Working Mathematician," Graduate Texts in Mathematics, vol. 5, Springer-Verlag, New York, 1971.

3. Samuel Eilenberg and Saunders Mac Lane, "General Theory of Natural Equivalences," Transactions of the American Mathematical Society, vol. 58, no. 2, pp. 231-294, 1945.

4. Daniel M. Kan, "Adjoint Functors," Transactions of the American Mathematical Society, vol. 87, no. 2, pp. 294-329, 1958.

5. Chris Heunen, Jamie Vicary, and Stefan Wolf, “Categories for Quantum Theory: An Introduction,” Oxford Graduate Texts, Oxford University Press, Oxford, 2018.

6. S. Eilenberg and J. C. Moore, “Adjoint Functors and Triples,” Proceedings of the Conference on Categorical Algebra, La Jolla, California, 1965, pp. 89-106.

7. Daniel M. Kan, “On Adjoints to Functors” (1958): In this paper, Kan further explored the theory of adjoint functors, focusing on the existence and uniqueness of adjoints. His work provided important insights into the fundamental aspects of adjoint functors and their role in category theory.

About the Author

Dean Young is a graduate student at New York University, where he studies mathematics.