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

Three Variations of the Whitehea

`IntCat`	`D(∞-Cat)`	$\vec{\Sigma}$	$\vec{\Omega}$	$\vec{\mathrm{P}}$	`InfPreShf`	`D(∞-Cat/C)`	$\vec{\sigma}$	$\vec{\omega}$	$\vec{\mathrm{p}}$
`IntGrpd`	`D(∞-Grpd)`	$\overleftrightarrow{\Sigma}$	$\overleftrightarrow{\Omega}$	$\overleftrightarrow{\mathrm{P}}$	`IntAct`	`D(∞-Grpd/G)`	$\overleftrightarrow{\sigma}$	$\overleftrightarrow{\omega}$	$\overleftrightarrow{\mathrm{p}}$
`IntGrp`	$\mathtt{D(\infty\text{-}Grpd_0)}$	Σ	Ω	P	$\mathtt{IntAct_0}$	$\mathtt{D(\infty\text{-}Grpd_0/G_0)}$	σ	ω	p

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

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

$$\forall(\mathrm{X{:}D(\infty\text{-}Grpd_0)}),\forall(\mathrm{Y{:}D(\infty\text{-}Grpd_0)}),\forall(\mathrm{f{:}X} \longrightarrow \mathrm{Y}),\forall(\mathrm{g{:}X} \longrightarrow \mathrm{Y}),(\forall(\mathrm{n{:}Nat}),(\pi_n\ \mathrm{f} = \pi_n\ \mathrm{g})) \to \mathrm{f} = \mathrm{g}$$

E. Dean Young and Jiazhen Xia

Plans to prove three variations of the
Whitehead theorem of homotopy groups in
Lean 4, with extensive use of Mathlib 4

We wish to acknowledge the collaborative efforts of E. Dean Young and Jiazhen Xia. Dean Young initially formulated the introduction with twelve goals, posting them on the Lean Zulip in August of 2023. Together the authors are pursuing these plans as a long term project.

1. Contents

The table of contents below reflects the tentative long-term goals of the authors, with the main goal the pursuit of the Whitehead theorem for a point-set model involving Mathlib's predefined homotopy groups.

$\vec{\omega}$ f : ∞-Grpd/D $\longrightarrow$ ∞-Grpd/C	The directed homotopy pullback functor
$\vec{\sigma}$ f : ∞-Grpd/C $\longrightarrow$ ∞-Grpd/D	Homotopy pushout with a point
$\vec{\pi}_n$: ∞-Grpd $\longrightarrow$ Set	The connected components functors

`Chapter 7:` The Whitehead Theorem for ∞-Groupoids

REP for ∞-groupoids	The cofibrant replacement functor for ∞-groupoids
HEP for ∞-groupoids	The homotopy extension property
Whitehead theorem (b)	A map `F : D(`∞`-Grpd).Hom E B` is determined by λ`(n:Nat),`$\vec{\pi}_n$ `F`.

`Chapter 8:` 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
$\vec{\mathrm{P}}$	The (remembrant derived) path space functor
$\vec{\mathrm{p}}$ C	The (remembrant derived) homotopy pullback functor

`Chapter 9:` The Groupoid Fixed Point Principals

The internal groupoid recognition principal	`D(`∞`-Grpd)` is deloopable internal categories in itself
The internal groupoid action recognition principal	`D(`∞`-Grpd/C)` is deloopable internal groupoid actions in itself

`Chapter 10:` The Category of Pairs of ∞-Groupoids

Pair Grpd	The category of pairs of ∞-groupoids
$\wedge$_(Pair ∞-Grpd), [,]_(Pair ∞-Grpd)	The cartesian closed structure on Pair ∞-Grpd
D(Pair ∞-Grpd)	The derived category of pairs of ∞-groupoids
$\wedge$_(D(Pair ∞-Grpd)), [,]_(D(Pair ∞-Grpd))	The cartesian closed structure on D(Pair ∞-Grpd)

`PART III:` ∞-Categories

`Chapter 11:` ∞`-Cat`

D(∞-Cat)	The derived category of ∞-categories
D(∞-Cat/C)	The derived category of ∞-categories over C
$\vec{\Omega}$: ∞-Cat $\longrightarrow$ ∞-Cat	The directed path space functor
$\vec{\Sigma}$: ∞-Cat $\longrightarrow$ ∞-Cat	The directed unbased suspension
$\vec{\omega}$ f : ∞-Cat/D $\longrightarrow$ ∞-Cat/C	The directed homotopy pullback functor
$\vec{\sigma}$ f : ∞-Cat/C $\longrightarrow$ ∞-Cat/D	The directed homotopy pushout
$\vec{\pi}_n$: ∞-Cat $\longrightarrow$ Set	The connected components functors

`Chapter 12:` The Whitehead Theorem for ∞-Categories

REP for ∞-categories	The cofibrant replacement functor for ∞-categories
HEP for ∞-categories	The directed homotopy extension property
Whitehead theorem (a)	A map `F : D(`∞`-Cat).Hom E B` is determined by λ`(n:Nat),`$\vec{\pi}_n$`F`.

`Chapter 13: Internal Categories and Internal Sheaves`

`IntCat Γ C`	The category of internal categories
`InfPreShf Γ C X`	The category of internal presheaves
The internal category principal	`f ×_(B) f` forms a component of an internal category
The internal presheaf principal	`f ×_(B) g` forms a component of an internal presheaf
$\vec{\mathrm{P}}$: D(∞-Cat) $\longrightarrow$ IntCat D(∞-Cat)	The (remembrant derived) directed path space functor
$\vec{\mathrm{p}}$ C : D(∞-Cat/C) $\longrightarrow$ InfPreShf D(∞-Cat/C)	The (remembrant derived) directed homotopy pullback functor

`Chapter 14:` The Categorical Fixed Point Principals

The internal category principal	`D(`∞`-Cat)` is equivalent to deloopable internal categories in itself
The internal sheaf principal	`D(`∞`-Cat/C)` is equivalent to deloopable internal presheaves in itself

`Chapter 15:` The Category of Pairs of ∞-Categories

Pair ∞-Cat	The category of pairs
$\wedge$_(Pair ∞-Cat), [,]_(Pair ∞-Cat)	The cartesian closed structure on Pair Grpd_0
D(Pair ∞-Cat)	The derived category of pairs
$\wedge$_(D(Pair ∞-Cat)), [,]_(D(Pair ∞-Cat))	The cartesian closed structure on D(Pair ∞-Cat)

2. Introduction

The main goals of this repository is to prove three variations of the Whitehead theorem and to establish three variations of the Puppe sequence. It is important that initial pull requests stemming from our work remain basic and accessible; we hope to make progress which is gradual and incremental.

Besides this goal, we have two others. 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),($\overleftrightarrow{\pi}_n$ F = $\overleftrightarrow{\pi}_n$ G)) $\to$ F = G, where $\overleftrightarrow{\pi}_n$ is notation for $\overleftrightarrow{\pi}$ 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 have stated these theorems in the above in an order reversed from the order of its implementation. This choice

We will use two models of each of the following categories in the theorems above:

(i) We model ∞-Cat : Cat firstly as the category of categories enriched over a convenient category of topological spaces, and secondly as the category of quasicategories.

(ii) We model ∞-Grpd : Cat firstly as a convenient category of topological spaces, and secondly as the category of Kan complexes.

(iii) We model ∞-Grpd$_0$: Cat firstly as the based connected objects of a convenient category of topological spaces, and secondly as the category of based connected Kan complexes.

This choice accords with the standard approach to the third theorem, in which one typically chooses both a combinatorial and point-set model, with the former featuring a geometric realization functor into the latter (`Mathlib` already has this).

We will make heavy use of `Mathlib 4`'s material on category theory, particularly their categories, functors, and natural transformations:

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

While the functors π_n occuring in the main theorems above are already defined in `Mathlib` for the desired point-set model, the functors $\vec{\pi}_n$ and $\overleftrightarrow{\pi}_n$ are not, and their definition will require great care. Here are their types:

(i) $\vec{\pi}_n$: Functor ∞-Cat Set

(ii) $\overleftrightarrow{\pi}_n$: Functor ∞-Grpd Set

(iii) π_n : Functor ∞-Grpd$_0$ Set

We may wish to modify these types out of convenience and to accord with the pre-existing functors π_n in `Mathlib 4`.

The existence of a base point makes π_n relatively straightforward to define, while $\vec{\pi}_n$ and $\overleftrightarrow{\pi}_n$ 'grow' as n does. We also form their derived functors:

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

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

(iii) D(π_n) : D(∞-Grpd$_0$) $\longrightarrow$ D(Set) $\simeq$ Set

In the course of the repository we will need the directed path space, path space, and loop space functors as well, which fit with the analogy formed by the Whitehead theorem and its two variations:

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

The third theorem (c), is the one from Whitehead's original papers.

With the choice of quasicategories as a combinatorial model, we hope to give good integration with Mathlib's existing features (though technically only the inner horns and simplices are defined, not even the category of quasicategories itself).

In the directed context, a homotopy between two maps in ∞-Cat⁄C consists of a sequence of compatible directed homotopies with the odd morphisms in the sequence 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`).

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 $H : \Delta^n \times \Delta^1 \longrightarrow Y$. The directed case requires an extra technical feature. 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.

We will define three different kinds of derived category:

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

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

1. For C : D(∞-Cat), a category D(∞-Cat/C) : Cat
2. For G : D(∞-Grpd), a category D(∞-Grpd/G) : Cat
3. For G_0 : D(∞-Grpd$_0$), a category D(∞-Grpd$_0$/G_0) : 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 $I := [0,1]$.

We will use six “internal” structures in addition to the standard structures in category theory:

1. IntCat : Cat → Cat
2. InfPreShf : (X : Cat) → (C : (IntCat X)) → Cat
3. IntGrpd : Cat → Cat
4. IntAct : (X : Cat) → (G : (IntGrpd X)) → Cat
5. IntGrp : Cat → Cat
6. IntAct_0 : (X : Cat) → (G_0 : (IntGrp X)) → Cat

The book “Galois theories” by Borceux and Janelidze deserves special mention as an inspiration for these internal structures. 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.

The six internal structures above arise here in relation 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 $D(\infty\text{-Cat}) \cong \mathrm{IntCat}\ D(\infty\text{-Cat})$ (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,-]), 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) $\omega\ (\mathbb{1}\ X) : \infty\text{-Grpd}_0/X_0 \longrightarrow \infty\text{-Grpd}_0/X_0$, the homotopy pullback with the base of X_0. $D(\omega\ (\mathbb{1}\ X))$ 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 exposition of G-principal bundles.

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

1. For C, D : D(∞-Cat), and f : C $\longrightarrow$ D, $\vec{\omega}$ f : D(∞-Cat/D) $\longrightarrow$ D(∞-Cat/C) (derived directed homotopy pullback)
2. For B, E : D(∞-Grpd), and f : E $\longrightarrow$ B, $\overleftrightarrow{\omega}$ f : D(∞-Grpd/B) $\longrightarrow$ D(∞-Grpd/E) (derived homotopy pullback)
3. For B_0, E_0 : D(∞-$Grpd_0$), and f : $E_0 \longrightarrow B_0$, ω f : D(∞-$Grpd_0/B_0$) $\longrightarrow$ D(∞-$Grpd_0/E_0$) (homotopy pullback with the base)

These six factored functors $\vec{P}$, $\overleftrightarrow{P}$, P : D(∞-$Grpd_0$), $\vec{p}$ ($\mathbb{1}$ C), $\overleftrightarrow{p}$ ($\mathbb{1}$ X), p are each fully faithful and produce categorical equivalences; we later construct functors $\vec{B}$, $\overleftrightarrow{B}$, B, $\vec{b}$, $\overleftrightarrow{b}$, b defined on the essential image of these six, which are inverse to them up to natural isomorphism.

We obtain six categorical equivalences witnessed by these twelve functors (along with twelve natural isomorphisms). Here are the types of $\vec{P}$, $\overleftrightarrow{P}$, P : D(∞-$Grpd_0$), $\vec{p}$ ($\mathbb{1}$ C), $\overleftrightarrow{p}$ ($\mathbb{1}$ X), p:

1. The directed path space, the path space, and loop space form components of the functors $\vec{P}$, $\overleftrightarrow{P}$, and P, which are valued in internal categories, internal groupoids, and internal groups respectively.
   (a) $\vec{P}$: D(∞-Cat) $\longrightarrow$ Cat D(∞-Cat)
   (b) $\overleftrightarrow{P}$: D(∞-Grpd) $\longrightarrow$ Grpd D(∞-Grpd)
   (c) P : D(∞-$Grpd_0$) $\longrightarrow$ Grp D(∞-Grpd) (see here)
2. The directed homotopy pullback, the homotopy pullback, and the homotopy pullback with the base form components of the functors Alg(Mon($\vec{\omega}$)), Alg(Mon($\overleftrightarrow{\omega}$)), and Alg(Mon(p)), respectively.
   (a) $\vec{p}$ ($\mathbb{1}$ C) : D(∞-Cat⁄C) $\longrightarrow$ InfPreShf D(∞-Cat⁄C) $\vec{P}$.obj C
   (b) $\overleftrightarrow{p}$ ($\mathbb{1}$ X) : D(∞-Grpd⁄X) $\longrightarrow$ IntAct D(∞-Grpd⁄X) $\overleftrightarrow{P}$.obj X
   (c) p ($\mathbb{1}$ X_0) : D(∞-$Grpd_0$⁄X_0) $\longrightarrow$ $IntAct_0$ D(∞-$Grpd_0$⁄X_0) P.obj X_0

Above, the functors $\vec{P}$, $\overleftrightarrow{P}$, P, $\vec{p}$, $\overleftrightarrow{p}$, and p feature $\vec{\Omega}$, $\overleftrightarrow{\Omega}$, Ω, $\vec{\omega}$, $\overleftrightarrow{\omega}$, and ω in their components, and can be related to them using constructions from Eilenberg-Moore theory.

The functors above will be defined as certain homotopy colimits, themselves certain coequilizers. On the condition that an internal category is internally filtered and internally cofiltered, we can further construct the $\vec{B}$.

We will make extensive use of Mathlib's bicategory of categories and material on simplicial sets. We further use Mathlib's pullbacks and categorical products, as well as their Eilenberg-Moore theory constructions. I'd like to extend my appreciation to

Scott Morison, Eric Wieser, Floris Van Doorn, and all the contributors who have put their efforts into creating these robust features for Mathlib 4.

Altogether, the project gets the following "periodic table" of 30 functors featured on the front cover:

`D(∞-Cat)`	$\vec{\Sigma}$	$\vec{\Omega}$	$\vec{\mathrm{P}}$	$\vec{\mathrm{B}}$	$\vec{\mathrm{E}}$	`D(∞-Cat/C)`	$\vec{\sigma}$	$\vec{\omega}$	$\vec{\mathrm{b}}$	$\vec{\mathrm{p}}$	$\vec{\mathrm{e}}$
`D(∞-Grpd)`	$\overleftrightarrow{\Sigma}$	$\overleftrightarrow{\Omega}$	$\overleftrightarrow{\mathrm{P}}$	$\overleftrightarrow{\mathrm{B}}$	$\overleftrightarrow{\mathrm{E}}$	`D(∞-Grpd/G)`	$\overleftrightarrow{\sigma}$	$\overleftrightarrow{\omega}$	$\overleftrightarrow{\mathrm{b}}$	$\overleftrightarrow{\mathrm{p}}$	$\overleftrightarrow{\mathrm{e}}$
`D(∞-Grpd`$_0$`)`	Σ	Ω	P	B	E	`D(∞-Grpd`$_0$`/G`$_0$`)`	σ	ω	b	p	e

Here are the names of the symbols featured above:

`Suspensional`	`Deductive`	`Remembrant`	`Delooping`	`Free`
$\vec{\Sigma}$ (Directed suspension)	$\vec{\Omega}$ (Directed path space)	$\vec{\mathrm{P}}$ (Remembrant derived directed path space)	$\vec{\mathrm{B}}$ (Classifying space for internal categories)	$\vec{\mathrm{E}}$
$\overleftrightarrow{\Sigma}$ (Suspensionoid)	$\overleftrightarrow{\Omega}$ (Path space)	$\overleftrightarrow{\mathrm{P}}$ (Remembrant derived path space)	$\overleftrightarrow{\mathrm{B}}$ (Classifying space for internal groupoids)	$\overleftrightarrow{\mathrm{E}}$
Σ (Suspension)	Ω (Loop space)	P (Remembrant derived loop space)	B (Classifying space for internal groups)	E
$\vec{\sigma}$	$\vec{\omega}$ (Directed homotopy pushout with a point)	$\vec{\mathrm{p}}$ (Remembrant derived directed homotopy pullback)	$\vec{\mathrm{b}}$ (Classifying space for internal presheaves)	$\vec{\mathrm{e}}$
$\overleftrightarrow{\sigma}$	$\overleftrightarrow{\omega}$ (Homotopy pushout with a point)	$\overleftrightarrow{\mathrm{p}}$ (Remembrant derived homotopy pullback)	$\overleftrightarrow{\mathrm{b}}$ (Classifying space for internal groupoid actions)	$\overleftrightarrow{\mathrm{e}}$
σ	ω (Homotopy fiber)	p (Remembrant derived homotopy fiber)	b (Classifying space for internal group actions)	e

The term "remembrant" in the above is not common terminology. It is intended to mean that 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 three-fold division of the project (three Whitehead theorems, three Puppe sequences, etc.) manifest while sticking to the standard notation for the established theorems (Σ, Ω, B, E). In the above, P could be said to stand for "(remembrant) path space" and p for "(remembrant) pullback", while at the same time this matches the theme that our capital letters reflect various internal structures and that their lower-case forms reflect the corresponding actions.

The mentioned "recognition principals", which identify inverses to the remembrant functors *on their essential image*, form important consequences of the three Whitehead theorems. All in all, there are twelve important theorems we want to 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_recognition_principal : Type`.

(IV) Define and inhabit the `internal_sheaf_recognition_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_recognition_principal : Type`.

(VIII) Define and inhabit the `internal_groupoid_action_recognition_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_recognition_principal : Type`.

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

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 D(P) • D(B) is naturally isomorphic to the identity functor on the essential image of D(P), and that D(B) • D(P) is naturally isomorphic to the identity functor on D(∞-Grpd$_0$)
4. The theorem that BG classifies G-principal bundles

These four established theorems and the eight novel ones form a good vignette for our project. Where previous approaches have considered comparisons between models such as the ones we have developed here, the goals above provide a unification which I would like to discuss in the section on pairs.

In the work that ensues, we plan to take an approach which establishes the known results before the original ones, taking advantage of the predefined π_n functors in `Mathlib 4` in the process. This decision will also help to take an approach which is more gradual and incremental, and to start with smaller pull requests.

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 a natural number `n` using the type `Nat` that comes with Lean 4.

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 p 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 `p` and `q`. `p` 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 great 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			
×	2A2F	\times	Product of types
→	2192	\rightarrow	Hom of types
⟨,⟩	27E8,27E9	\langle,\rangle	Product term introduction
↦	21A6	\mapsto	Hom term introduction
∧	2227	\wedge	Conjunction
∨	2228	\vee	Disjunction
∀	2200	\forall	Universal quantification
∃	2203	\exists	Existential quantification
¬	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
∘	2218	\circ	Composition
Bicategories			
•	2022	\smul	Horizontal composition of objects
Adjunctions			
⇄	21C4	\rightleftarrows	Adjunctions
⇆	21C6	\leftrightarrows	Adjunctions
.	1BC94		Right adjoints
·	0971		Left adjoints
⊣	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			
∼	223C	\sim	Homotopies
≃	2243	\equiv	Equivalences
≅	2245	\cong	Isomorphisms
⊥	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 !, ¡, ., ˙, →, and ↔ 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.

The conceptual difference between the first, second, and third Whitehead theorems.

PART 1: BASED CONNECTED ∞-GROUPOIDS

Chapter 1: ∞-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.

5. Ω

6. ω

7. π_0

Chapter 2: 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$_n$: $\Delta^1 \times$ X$_n$ $\longrightarrow$ Y between $f|_{X_n}, g|_{X_n}$: X$_n$ $\longrightarrow$ Y, we construct an extension of the homotopy equivalence $\Delta^1 \times X_{n+1} \longrightarrow$ Y.

8. HEP for based connected ∞-groupoids

This

9. The Whitehead theorem

Here we show the Whitehead theorem proper.

Chapter 3: Internal Groups and Internal Group Actions

10. Grp_(Γ)

Lean 7

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

Lean 8

Lean 9

Lean 10

Lean 11

Lean 12

Lean 13

Lean 14

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

Lean 15

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

11. 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 16

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

Lean 17

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

Lean 18

Lean 19

Lean 20

Lean 21

Lean 22

12. The Internal Group Principal

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

13. 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.

14. P

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

15. 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 4: The Category of Pairs

PART 2: ∞-GROUPOIDS

Chapter 5: ∞-Grpd

16. Ω

Our choice of symbols refects our choice of three variations of the Whitehead theorem and three Puppe sequences. $\overleftrightarrow{\Omega}$, the analogue of loop space, is the internal hom functor [I,-] : ∞-Grpd $\longrightarrow$ ∞-Grpd. 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 $f_1 \bullet g_1 \bullet f_2 \bullet g_2 \bullet \cdots \bullet f_n \bullet g_n$, 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.

17. ω

$\vec{\Omega}$ is to internal categories as $\overleftrightarrow{\omega}$ is to internal G-actions. It is also called directed homotopy pullback. These functors will later be used to produce functors $\overleftrightarrow{\mathrm{P}} : \mathrm{D}(\infty\text{-Grpd}) \longrightarrow \mathrm{IntCat}\ \mathrm{D}(\infty\text{-Grpd})$ and $\overleftrightarrow{\mathrm{p}} : \mathrm{D}(\infty\text{-Grpd}/\mathrm{C}) \longrightarrow \mathrm{InfPreShf}\ (\overleftrightarrow{\mathrm{P}}\ \mathrm{G})\ \mathrm{D}(\infty\text{-Grpd}/\mathrm{G})$.

18. π_n

The mentioned functors $\overleftrightarrow{\pi}_n$ are designed with both Whitehead theorem (a) and Puppe sequence (a) in mind.

Chapter 6: The Whitehead Theorem for ∞-Groupoids

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 $\mathcal{D}(\infty\text{-Cat})$, at first, and in a formal way, so as to create a category whose object component $\mathcal{D}(\infty\text{-Cat}).\alpha$ matches the object component $\infty\text{-Cat}.\alpha$ 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" $\mathrm{R} : \infty\text{-Cat}.\alpha \longrightarrow \infty\text{-Cat}.\alpha$ ($.\alpha$ 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 $\mathrm{F} : \infty\text{-Cat} \longrightarrow \infty\text{-Cat}$. $\mathrm{D(F)} : \mathrm{D}(\infty\text{-Cat}) \longrightarrow \mathrm{D}(\infty\text{-Cat})$ should be a categorical equivalence, and that is what we will do.

2. (HEP) For the object R X, demonstrate that any $\mathrm{F,G} : (\mathrm{R\ X}) \longrightarrow \mathrm{Y}$ such that $\forall(\mathrm{n{:}Nat}),(\vec{\pi}_n\ \mathrm{F} = \vec{\pi}_n\ \mathrm{G})$, 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⍰ : $\Delta^1 \times (\mathrm{R\ X}) \longrightarrow \mathrm{Y}$, 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 recognition 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 $\vec{\Omega}^{\mathrm{n}}$. 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: $\vec{\Omega}^{\mathrm{n}}$ X which are homotopic by a homotopy which restricts to a constant along the face maps f$\boxed{?}$: $\vec{\Omega}^{n-1}$ 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 : $\Delta^{1} \times (\partial((\Delta^{1})^{\mathrm{n}}))$ $\longrightarrow (\Delta^{1})^{n+1}$ by the projection map $\Delta^{1} \times (\partial((\Delta^{1})^{\mathrm{n}})) \longrightarrow \partial((\Delta^{1})^{\mathrm{n}})$

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 : $\Delta^{1} \times (\partial(\Delta^{\mathrm{n}})) \longrightarrow (\Delta^{1}) \times \Delta^{\mathrm{n}}$ by the projection map $\Delta^{1} \times (\partial((\Delta^{1})^{\mathrm{n}})) \longrightarrow$ $\partial(\Delta^{\mathrm{n}})$

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 $\partial((\Delta^{1})^{\mathrm{n}}) \longrightarrow$ $(\Delta^{1})^{\mathrm{n}}$

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

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 $(\Delta^1)^{n-1} \longrightarrow \Delta^{1n}$, 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` $\longrightarrow$ `(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.

19. 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`

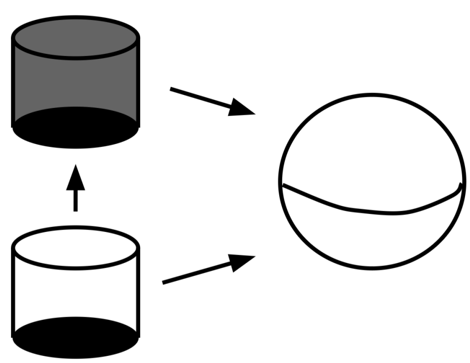
20. HEP

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 : $\partial\Delta^2 \longrightarrow Y$, 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.

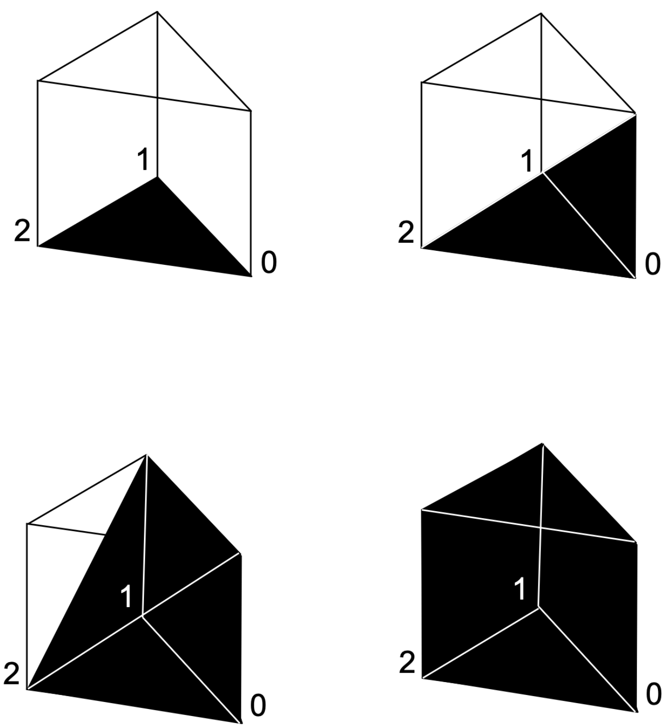


Prism Filling (PF) Let Y be a quasicategory, and let f, g : $\partial\Delta^n \longrightarrow Y$. A homotopy $h : \partial\Delta^n \times \Delta^1 \longrightarrow Y$ between f, g : $\partial\Delta^n \longrightarrow Y$ extends to a map $H : \Delta^n \times \Delta^1 \longrightarrow Y$;

this follows from the condition that Y be a quasicategory. 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.

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 ($\vec{\Omega}^n$ X)– one which amounts to taking an equivalence relation by paths in $\vec{\Omega}^n$ X which restrict to constant paths along the face maps f⍰ : $\vec{\Omega}^{n-1}$ X $\longrightarrow$ $\vec{\Omega}^n$ 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 $\vec{\Omega}^n$ 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 $\vec{B}$ and $\vec{E}$, 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 $\vec{\Omega}^n$ 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.

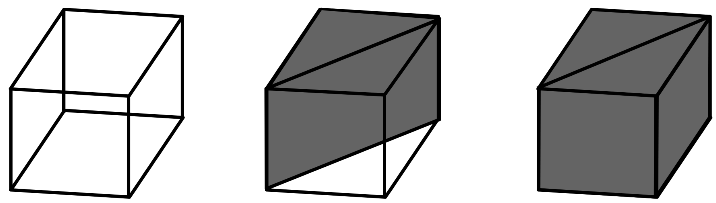
This means that we want directed box filling in addition to directed prism filling (but which also uses directed prism filling in its proof).

Box Filling (BF) Let Y be a quasicategory, and let f, g : $\partial\Delta^n \longrightarrow$ Y. 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; this follows from the condition that Y be a quasicategory. 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 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 $\partial\Delta^n \times \Delta^1$. 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.

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 BF 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:

21. The Whitehead Theorem for ∞-Cat

The HEP in the last

..H(-,1) and g match on $\partial\Delta^{\mathtt{n}}$, producing a map $\mathrm{f} : \mathrm{X} \longrightarrow \mathrm{Y}$, where X consists of two copies of $\Delta^{\mathtt{n}}$ glued together at the boundary. Consider a space X' formed as a quotient of $\Delta^{\mathtt{n}} \times \Delta^{1}$ by $\partial\Delta^{\mathtt{n}} \times \Delta^{1}$. There is a map $\phi : \mathrm{X} \longrightarrow \mathrm{X'}$. An induction hypothesis on f and g involving π_n ensures that the aparent map $\mathrm{X} \longrightarrow \mathrm{Y}$ lifts along ϕ, producing a map from $\Delta^{\mathtt{n}} \times \Delta^{1}$ which is constant on $\partial\Delta^{\mathtt{n}} \times \Delta^{1}$. 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 7: Internal Groupoids and Internal Groupoid Actions

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

`Section`	`Description`
`IntCat Γ : Cat`	Internal categories
`InfPreShf 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
$\vec{P}$ C : IntCat D(∞-Cat)	$\vec{\Omega}$ C forms a component of an internal category
$\vec{p}$ ($\mathbb{1}$ C) D : InfPreShf D(∞-Cat/C) ($\vec{P}$ C)	$\vec{\omega}$ ($\mathbb{1}$ C) D forms a component of an internal C-presheaf

22. IntCat Γ

In this chapter I define an internal category. Internal categories are most commonly defined on categories with enough pullbacks, but here I may also like to keep in mind that it is valuable to be able to iterate `IntCat` in the way of composition.

Lean 23

```
-- 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 24

```
-- 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 25

```
-- 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 26

```
-- 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 27

```
-- 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 28

```
-- 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 29

```
-- 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 30

```
/-
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 31

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

23. InfPreShf 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 32

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

Lean 33

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

Lean 34

```
-- 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 35

```
-- 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 36

```
-- proving the first identity law for internal
 ↪  functors
/-
def Shf... (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 37

```
-- 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 38

```
-- 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 39

```
/-
def InfPreShf (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 40

```
/-
notation : 2000 "Shf_(" Γ ")" => InfPreShf 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.

24. 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 in which this pullback exists. In fact, the most general form of the theorem works for a noncommutative analogue of pullback.

25. 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.

26. 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 41

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

Lean 42

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

Lean 43

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

Lean 44

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

Lean 45

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

Lean 46

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

Lean 47

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

Lean 48

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

Lean 49

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

Lean 50

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

Lean 51

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

Lean 52

```
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 53

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

27. 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 54

```
-- 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) (InfPreShf Γ) (P_(Γ) E B
↪  f)
-/
```

Chapter 8: The Category of Pairs of ∞-Groupoids

PART 3: ∞-CATEGORIES

Chapter 9: ∞-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.

28. 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 $\longrightarrow$ ∞-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 $f_1 \bullet g_1 \bullet f_2 \bullet g_2 \bullet \cdots \bullet f_n \bullet g_n$, 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.

29. ω

$\vec{\Omega}$ is to internal categories as $\vec{\omega}$ is to internal C-presheaves. 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 $\vec{\mathrm{p}} : \mathrm{D}(\infty\text{-Cat}/\mathrm{C}) \longrightarrow \mathrm{InfPreShf}\,(\vec{\mathrm{P}}\ \mathrm{C})\ \mathrm{D}(\infty\text{-Cat}/\mathrm{C})$.

30. π_n

The mentioned functors $\vec{\pi}_n$ are designed with both Whitehead theorem (a) and Puppe sequence (a) in mind.

Chapter 10: 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 $\mathcal{D}(\infty\text{-Cat})$, at first, and in a formal way, so as to create a category whose object component $\mathcal{D}(\infty\text{-Cat}).\alpha$ matches the object component $\infty\text{-Cat}.\alpha$ 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" $\mathrm{R} : \infty\text{-Cat}.\alpha \longrightarrow \infty\text{-Cat}.\alpha$ ($.\alpha$ 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 $\mathrm{F} : \infty\text{-Cat} \longrightarrow \infty\text{-Cat}$. $\mathrm{D(F)} : \mathrm{D}(\infty\text{-Cat}) \longrightarrow \mathrm{D}(\infty\text{-Cat})$ should be a categorical equivalence, and that is what we will do.

2. (HEP) For the object R X, demonstrate that any $\mathrm{F,G} : (\mathrm{R\ X}) \longrightarrow \mathrm{Y}$ such that $\forall(\mathrm{n{:}Nat}),(\vec{\pi}_n\ \mathrm{F} = \vec{\pi}_n\ \mathrm{G})$, 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□ : $\Delta^1 \times (\mathrm{R\ X}) \longrightarrow \mathrm{Y}$, 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 $[\Delta^1,\text{-}]$ (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 recognition 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 $\vec{\Omega}^{\mathrm{n}}$. 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: $\vec{\Omega}^{\mathrm{n}}$ 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 : $\Delta^1 \times (\partial((\Delta^1)^{\mathrm{n}}))$ $\longrightarrow (\Delta^1)^{n+1}$ by the projection map $\Delta^1 \times (\partial((\Delta^1)^{\mathrm{n}})) \longrightarrow \partial((\Delta^1)^{\mathrm{n}})$

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 : $\Delta^1 \times (\partial(\Delta^{\mathrm{n}})) \longrightarrow (\Delta^1) \times \Delta^{\mathrm{n}}$ by the projection map $\Delta^1 \times (\partial((\Delta^1)^{\mathrm{n}})) \longrightarrow$ $\partial(\Delta^{\mathrm{n}})$

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 $\partial((\Delta^1)^{\mathrm{n}}) \longrightarrow$ $(\Delta^1)^{\mathrm{n}}$

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

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 $(\Delta^1)^{n-1} \longrightarrow \Delta^{1n}$, 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` $\longrightarrow$ `(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.

31. 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`

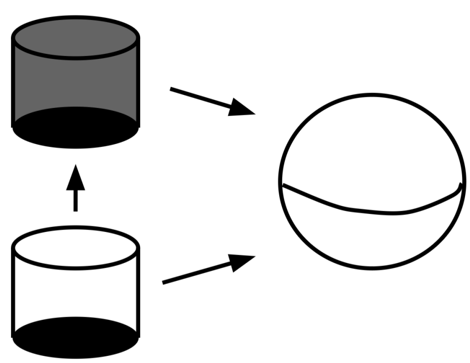
32. HEP

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 : $\partial\Delta^2 \longrightarrow$ 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 quasi-category 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.

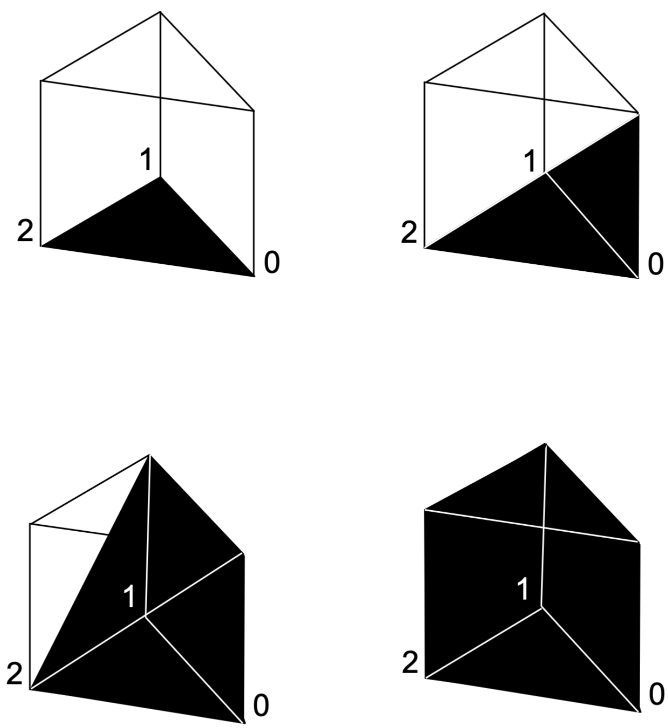


Prism Filling (PF) Let Y be a quasicategory, and let f, g : $\partial\Delta^n \longrightarrow$ Y. A homotopy h : $\partial\Delta^n \times \Delta^1 \longrightarrow$ Y between f, g : $\partial\Delta^n \longrightarrow$ Y extends to a map H : $\Delta^n \times \Delta^1 \longrightarrow$ Y;

this follows from the condition that Y be a quasicategory. 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.

Directed prism filling may combine fruitfully with the yoneda lemma and/or the fact that simplicial sets are determined by the sets $[\Delta^n,X]$ 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 ($\vec{\Omega}^n$ X)– one which amounts to taking an equivalence relation by paths in $\vec{\Omega}^{n}$ X which restrict to constant paths along the face maps f? : $\vec{\Omega}^{n-1}$ X $\longrightarrow \vec{\Omega}^{n}$ 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 $\vec{\Omega}^{n}$ 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 $\vec{\Omega}^{n}$ 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.

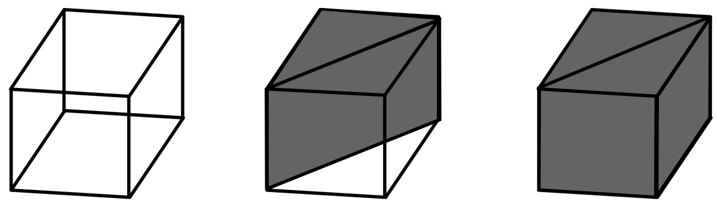
This means that we want directed box filling in addition to directed prism filling (but which also uses directed prism filling in its proof).

Box Filling (BF) Let Y be a quasicategory, and let f, g : $\partial\Delta^n \longrightarrow$ Y. 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; this follows from the condition that Y be a quasicategory. 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 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 $\partial\Delta^n \times \Delta^1$. 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.

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 BF 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:

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 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 $\partial\Delta^n \times \Delta^1$. 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 11: 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 category. 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
`InfPreShf 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 : InfPreShf D(∞-Cat/C) ($\vec{\mathrm{P}}$ C)	$\vec{\omega}$ ($\mathbb{1}$ C) D forms a component of an internal C-presheaf

33. IntCat Γ

In this chapter I define an internal category. Internal categories are most commonly defined on categories with enough pullbacks, but here I may also like to keep in mind that it is valuable to be able to iterate `IntCat` in the way of composition.

Lean 55

```
-- 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 56

```
-- 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 57

```
-- 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 58

```
-- 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 59

```
-- 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 60

```
-- 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 61

```
-- 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 62

```
/-
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 63

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

34. InfPreShf 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 64

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

Lean 65

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

Lean 66

```
-- 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 67

```
-- 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 68

```
-- proving the first identity law for internal
 ↪  functors
/-
def Shf... (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 69

```
-- 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 70

```
-- 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 71

```
/-
def InfPreShf (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 72

```
/-
notation : 2000 "Shf_(" Γ ")" => InfPreShf 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.

35. 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 in which this pullback exists. In fact, the most general form of the theorem works for a noncommutative analogue of pullback.

36. 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.

37. 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 73

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

Lean 74

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

Lean 75

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

Lean 76

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

Lean 77

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

Lean 78

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

Lean 79

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

Lean 80

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

Lean 81

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

Lean 82

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

Lean 83

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

Lean 84

```
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 85

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

38. 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 86

```
-- 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) (InfPreShf Γ) (P_(Γ) E B
↪  f)
-/
```

Chapter 12: The Category of Pairs of ∞-Categories

Bibliography

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.

Lectures, Videos, and Stackexchange questions:

1. `https://www.youtube.com/watch?v=Ob9tOgWumPI`
2. `https://www.youtube.com/watch?v=xYenPIeX6MY`
3. `https://mathoverflow.net/questions/5901/do-the-signs-in-puppe-sequences-matter`

Relevant discussions on the `Lean 4` Zulip:

1.

Ideas for future applications:

1. `https://arxiv.org/pdf/2206.13563.pdf`

About the Author

Dean Young is a master's student at New York University, where he studies mathematics.

About the Author

Jiazhen Xia is a graduate student at Zhejiang University, where he studies computer science.