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

The Whitehead Theorem and Two Variations

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

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

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

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.

Cubical Complexes	...
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 7: The Category of Pairs of ∞-Groupoids	
...	...
REP for pairs of ∞-groupoids	The replacement functor on ∞-Grpd
HEP for pairs of ∞-groupoids	The homotopy extension property for ∞-Grpd
The Whitehead theorem for pairs of ∞-groupoids	...
Chapter 8: Internal and Operadic Groupoids and their Actions	
InternalGroupoid Γ	The category of internal groupoids in Γ
InternalGroupoidAction Γ G	The category of internal G-actions in Γ
OperadicGroupoid Γ	The category of operadic groupoids
OperadicGroupoidAction Γ	The category of operadic groupoid actions
The P and O Functors	P = ? ● O
The p and o Functors	p = ? ● o
PART III: ∞-Categories	
Chapter 9: ∞-Cat	
D(∞-Cat)	The derived category of ∞-categories
D(∞-Cat/C)	The derived category of ∞-categories over C
$\vec{\Omega} : \infty\text{-Cat} \longrightarrow \infty\text{-Cat}$	The directed path space functor
$\vec{\Sigma} : \infty\text{-Cat} \longrightarrow \infty\text{-Cat}$	The directed unbased suspension
$\vec{\omega}\, f : \infty\text{-Cat/D} \longrightarrow \infty\text{-Cat/C}$	The directed homotopy pullback functor
$\vec{\sigma}\, f : \infty\text{-Cat/C} \longrightarrow \infty\text{-Cat/D}$	The directed homotopy pushout
$\vec{\pi}_n : \infty\text{-Cat} \longrightarrow \text{Set}$	The connected components functors
Chapter 10: The Whitehead Theorem for ∞-Categories	
Directed Cubical Complexes	...
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 11: The Category of Pairs of ∞-Categories	
REP for pairs of ∞-groupoids	The replacement functor on ∞-Grpd
HEP for pairs of ∞-groupoids	The homotopy extension property for ∞-Grpd
The Whitehead theorem for pairs of ∞-groupoids	...
Chapter 12: Internal and Operadic Categories and their Sheaves	
InternalCategory : ???	The category of internal categories
InternalPresheaf : ???	The category of internal presheaves
OperadicCategory : Cat → Cat	The category of operadic categories
OperadicPresheaf : Cat → Cat	The category of operadic presheaves
The P and O Functors	P = ? ● O
The p and o Functors	p = ? ● o

2. Introduction

The main goal of this repository is to prove the Whitehead theorem in Lean 4 using Mathlib 4's homotopy groups. Two other subsequent goals are to state and prove two variations of the Whitehead theorem. 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) $\forall$(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.

(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 ∞-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.

Here are a few comments on these theorems:

1. The first Whitehead theorem reflects the known one, which dates back to a paper that can be found in defined on Mathlib 4's homotopy groups, which make use of cubes I^{n} and their boundaries $\partial\mathrm{I}^{\mathrm{n}}$. There is also a fourth Whitehead theorem concerning ∞-Cat$_0$, but which is ommited; it is less general than the third while the first is present for its use of Mathlib 4's predefined π_n's.
2. The unicode symbol $\longrightarrow$ (U+2B62) reflects a notation not present in Mathlib 4, and which we will make local or scoped.
3. The three theorems above are stated in an order reversed from the order of their implementation.

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 Mathlib 4's topological spaces, and secondly as the category of quasicategories.

(ii) We model ∞-Grpd : Cat firstly as a Mathlib 4's topological spaces, and secondly as the category of Kan complexes.

(iii) We model ∞-Grpd$_0$: Cat firstly as the based connected objects using Mathlib 4's 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.

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`)
2. Functors (see Mathlib's `Functor C D`)
3. Natural transformations (see Mathlib's `NatTrans F G`)
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) $\mathrm{D}(\vec{\pi}_n) : \mathrm{D}(\infty\text{-Cat}) \longrightarrow \mathrm{D}(\mathrm{Set}) \simeq \mathrm{Set}$

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

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

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. $\vec{\Omega} : \infty\text{-Cat} \longrightarrow \infty\text{-Cat}$ sends

2. $\vec{\Omega} : \infty$-Grpd $\longrightarrow \infty$-Grpd is ...

3. Ω is the loop space functor

We develop two models of $(\infty,1)$-categories, one combinatorial in nature and another point-set in nature.

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

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 (strict) "internal" structures in addition to the standard structures in category theory:

(i) InternalCategory : Cat $\rightarrow$ Cat

(ii) InternalPresheaf : (X : Cat) $\rightarrow$ (C : (InternalCategory X)) $\rightarrow$ Cat

(iii) InternalGroupoid : Cat $\rightarrow$ Cat

(iv) InternalGroupoidAction : (X : Cat) $\rightarrow$ (G : (InternalGroupoid X)) $\rightarrow$ Cat

(v) InternalGroup : Cat $\rightarrow$ Cat

(vi) InternalGroupAction : (X : Cat) $\rightarrow$ (G_0 : (InternalGroup X)) $\rightarrow$ Cat

The book "Galois theories" by Borceux and Janelidze contains the internal structures (iii), (iv), (v), and (vi), and the first two internal structures have fewer entries. 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.

Some previous work done on these structures can be found at the thread here.

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 $D(\infty\text{-Cat}) \cong$ InternalCategory $D(\infty\text{-Cat})$ (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) $\omega\,(\mathbb{1}\ X) : \infty\text{-Grpd}_0/X_0 \longrightarrow \infty\text{-Grpd}_0/X_0$, 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 $\vec{\omega}\,(\mathbb{1}\ C)$, $\vec{\vec{\omega}}\,(\mathbb{1}\ X)$, and $\omega\,(\mathbb{1}\ C)$ in the above ensue from a more general construction:

1. For $C, D : D(\infty\text{-Cat})$, and $f : C \longrightarrow D$, $\vec{\omega}\, f : D(\infty\text{-Cat}/D) \longrightarrow D(\infty\text{-Cat}/C)$ (derived directed homotopy pullback)
2. For $B, E : D(\infty\text{-Grpd})$, and $f : E \longrightarrow B$, $\vec{\vec{\omega}}\, f : D(\infty\text{-Grpd}/B) \longrightarrow D(\infty\text{-Grpd}/E)$ (derived homotopy pullback)
3. For $B_0, E_0 : D(\infty\text{-Grpd}_0)$, and $f : E_0 \longrightarrow B_0$, $\omega\, f : D(\infty\text{-Grpd}_0/B_0) \longrightarrow D(\infty\text{-Grpd}_0/E_0)$ (homotopy pullback with the base)

These six factored functors $\vec{P}$, $\vec{\vec{P}}$, $P : D(\infty\text{-Grpd}_0)$, $\vec{p}\,(\mathbb{1}\ C)$, $\vec{\vec{p}}\,(\mathbb{1}\ X)$, p are each fully faithful and produce categorical equivalences; we later construct functors $\vec{B}$, $\vec{\vec{B}}$, B, $\vec{b}$, $\vec{\vec{b}}$, 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 $\vec{P}$, $\vec{\vec{P}}$, $P : D(\infty\text{-Grpd}_0)$, $\vec{p}\,(\mathbb{1}\ C)$, $\vec{\vec{p}}\,(\mathbb{1}\ X)$, p:

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

(b) $\overleftrightarrow{\mathrm{p}}\,(\mathbb{1}\ \mathrm{X}) : \mathrm{D}(\infty\text{-Grpd}/\mathrm{X}) \longrightarrow \mathrm{InternalGroupoidAction}\ \mathrm{D}(\infty\text{-Grpd}/\mathrm{X})\ \overleftrightarrow{\mathrm{P}}.\mathrm{obj}\ \mathrm{X}$

(c) $\mathrm{p}\,(\mathbb{1}\ \mathrm{X}_0) : \mathrm{D}(\infty\text{-Grpd}_0/\mathrm{X}_0) \longrightarrow \mathrm{InternalGroupAction}\ \mathrm{D}(\infty\text{-Grpd}_0/\mathrm{X}_0)\ \mathrm{P}.\mathrm{obj}\ \mathrm{X}_0$

Above, the functors $\vec{\mathrm{P}}$, $\overleftrightarrow{\mathrm{P}}$, P, $\vec{\mathrm{p}}$, $\overleftrightarrow{\mathrm{p}}$, 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.

Twelve Structures			
Strict		Lax	
Unitial	Actional	Unitial	Actional
InternalCategory	InternalPresheaf	OperadicCategory	OperadicPresheaf
InternalGroupoid	InternalGroupoidAction	OperadicGroupoid	OperadicGroupoidAction
InternalGroup	InternalGroupAction	OperadicGroup	OperadicGroupAction

$\vec{\Omega} : \infty\text{-Cat} \longrightarrow \infty\text{-Cat}$	$\vec{\omega} : (\mathrm{C} : \infty\text{-Cat}) \to (\mathrm{D} : \infty\text{-Cat}) \to (\mathrm{F} : \infty\text{-Cat.hom C D}) \to (\infty\text{-Cat}/\mathrm{D}) \longrightarrow \infty\text{-Cat}$
$\vec{\mathrm{O}} : \infty\text{-Cat} \longrightarrow \mathrm{OperadicCategory}\ \infty\text{-Cat}$	$\vec{\mathrm{o}} : (\mathrm{C} : \infty\text{-Cat}) \to (\mathrm{D} : \infty\text{-Cat}) \to (\mathrm{F} : \infty\text{-Cat.hom C D}) \to (\infty\text{-Cat}/\mathrm{D}) \longrightarrow$ Operadic
$\vec{\mathrm{P}} : \infty\text{-Cat} \longrightarrow \mathrm{InternalCategory}\ \mathrm{D}(\infty\text{-Cat})$	$\vec{\mathrm{p}} : (\mathrm{C} : \infty\text{-Cat}) \to (\mathrm{D} : \infty\text{-Cat}) \to (\mathrm{F} : \infty\text{-Cat.hom C D}) \to (\infty\text{-Cat}/\mathrm{D}) \longrightarrow$ InternalP
$\overleftrightarrow{\Omega} : \infty\text{-Grpd} \longrightarrow \infty\text{-Grpd}$	$\overleftrightarrow{\omega} : (\mathrm{X} : \infty\text{-Grpd}) \to (\mathrm{Y} : \infty\text{-Grpd}) \to (\mathrm{F} : \infty\text{-Cat.hom X Y}) \to (\infty\text{-Grpd}/\mathrm{Y}) \longrightarrow \infty$-G
$\overleftrightarrow{\mathrm{O}} : \infty\text{-Grpd} \longrightarrow \mathrm{OperadicGroupoid}\ \infty\text{-Grpd}$	$\overleftrightarrow{\mathrm{o}} : (\mathrm{X} : \infty\text{-Grpd}) \to (\mathrm{Y} : \infty\text{-Grpd}) \to (\mathrm{F} : \infty\text{-Cat.hom X Y}) \to (\infty\text{-Grpd}/\mathrm{Y}) \longrightarrow$ Oper
$\overleftrightarrow{\mathrm{P}} : \infty\text{-Grpd} \longrightarrow \mathrm{InternalGroupoid}\ \mathrm{D}(\infty\text{-Grpd})$	$\overleftrightarrow{\mathrm{p}} : (\mathrm{X} : \infty\text{-Grpd}) \to (\mathrm{Y} : \infty\text{-Grpd}) \to (\mathrm{F} : \infty\text{-Cat.hom C D}) \to (\infty\text{-Grpd}/\mathrm{Y}) \longrightarrow$ Inter
$\Omega : \infty\text{-Grpd}_{-1} \longrightarrow \infty\text{-Grpd}_{-1}$	$\omega : (\mathrm{X}_{-1} : \infty\text{-Grpd}_{-1}) \to (\mathrm{Y}_{-1} : \infty\text{-Grpd}_{-1}) \to (\mathrm{F} : \infty\text{-Grpd}_{-1}.\mathrm{hom}\ \mathrm{X}_{-1}\ \mathrm{Y}_{-1}) \to ($
$\mathrm{O} : \infty\text{-Grpd}_{-1} \longrightarrow \mathrm{OperadicGroup}\ \infty\text{-Grpd}_{-1}$	$\mathrm{o} : (\mathrm{X}_{-1} : \infty\text{-Grpd}_{-1}) \to (\mathrm{Y}_{-1} : \infty\text{-Grpd}_{-1}) \to (\mathrm{F} : \infty\text{-Grpd}_{-1}.\mathrm{hom}\ \mathrm{X}_{-1}\ \mathrm{Y}_{-1}) \to ($
$\mathrm{P} : \infty\text{-Grpd}_{-1} \longrightarrow \mathrm{InternalGroup}\ \mathrm{D}(\infty\text{-Grpd}_{-1})$	$\mathrm{p} : (\mathrm{X}_{-1} : \infty\text{-Grpd}_{-1}) \to (\mathrm{Y}_{-1} : \infty\text{-Grpd}_{-1}) \to (\mathrm{F} : \infty\text{-Grpd}_{-1}.\mathrm{hom}\ \mathrm{X}_{-1}\ \mathrm{Y}_{-1}) \to ($

3. 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 $^{!}$, $^{¡}$, $_{.}$, $^{\cdot}$, $^{\rightarrow}$, and $^{\leftrightarrow}$ do not have VSCode shortcuts.

4. 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)
```

Here are some introductions to Lean 4 and Mathlib 4:

1. The tutorial here gives an introduction to using the dependent type theory in Lean.

2.

PART 1: BASED CONNECTED ∞-GROUPOIDS

In this first section we prove the standard Whitehead theorem.

Chapter 1: ∞-Grpd$_0$

Implementation Progress

Lean 7

```lean
/-- A relative CW-complex contains an expanding
↪  sequence of subspaces `sk i`
(called the `i`-skeleta) for `i   -1`, where `sk
↪  (-1)` is an arbitrary topological space,
isomorphic to `A`, and each `sk (n+1)` is obtained
↪  from `sk n` by attaching (n+1)-disks. -/
structure RelativeCWComplex (A : TopCat) where
  /-- Skeleta -/
  sk : ℤ → TopCat
  /-- A is isomorphic to the (-1)-skeleton. -/
  iso_sk_neg_one : A ≅ sk (-1)
  /-- The (n+1)-skeleton is obtained from the
   ↪  n-skeleton by attaching (n+1)-disks. -/
  attach_cells : (n : ℤ) → CWComplex.AttachCells (sk
   ↪  n) (sk (n + 1)) n

/-- A CW-complex is a relative CW-complex whose
↪  (-1)-skeleton is empty. -/
abbrev CWComplex := RelativeCWComplex (TopCat.of
↪  Empty)
```

Lean 8

```
/-- The topology on a relative CW-complex -/
def toTopCat {A : TopCat} (X : RelativeCWComplex A) :
↪ TopCat :=
  Limits.colimit (colimitDiagram X)

instance : Coe CWComplex TopCat where coe X :=
↪ toTopCat X
```

Lean 9

```
def IsCWComplex (X : TopCat) : Prop := ∃ Y :
↪ CWComplex, Nonempty (↑Y ≅ X)

def CWComplexCat := FullSubcategory IsCWComplex
```

Writing Progress

Here we define CW-complexes, as well as relative CW-complexes, and also the derived categories $D(\infty\text{-Grpd}_0)$ of connected based ∞-groupoids and $D(\infty\text{-Grpd}_0/G_0)$, made from CW-complexes.

5. $\mathtt{D}(\infty\text{-}\mathtt{Grpd}_0)$

Symbol	Unicode	VSCode shortcut	Use
Lean's Kernel			
$\times$	2A2F	`\times`	Product of types
$\rightarrow$	2192	`\rightarrow`	Hom of types
$\dashv$	22A3	`\dashv`	The condition that two functors are adjoint
?,¿	003F, 00BF	`?,\?`	The corresponding (co)monad of an adjunction
$\sim$	223C	`\sim`	Homotopies

Lean 10

6. $D(\infty\text{-}\mathtt{Grpd}_0/X_0)$

The derived category of based connected ∞-groupoids over X_0.

7. $\Omega : \infty\text{-}\mathtt{Grpd}_0 \longrightarrow \infty\text{-}\mathtt{Grpd}$

8. $\Sigma : \infty\text{-}\mathtt{Grpd}_0 \longrightarrow \infty\text{-}\mathtt{Grpd}_0$

The based suspension functor

9. $\omega f : \infty\text{-}\mathtt{Grpd}/D_0 \longrightarrow \infty\text{-}\mathtt{Grpd}/C_0$

The homotopy fiber

$$10.\ \sigma\ f : \infty\text{-}\mathtt{Grpd}_0/C_0 \longrightarrow \infty\text{-}\mathtt{Grpd}_0/D_0$$

Based homotopy pushout

11. $\pi_n : \infty\text{-}\mathtt{Grpd}_0 \longrightarrow \mathtt{Set}$

The connected components functors

Chapter 2: The Whitehead Theorem

The proof of the Whitehead theorem divides into REP (replacement for based connected ∞-groupoids X : ∞-Grpd$_0$) and HEP (the homotopy extension property for weak equivalent maps of based ∞-groupoids). The replacement functor ∞-$\texttt{Grpd}_0$ can be constructed using globular sets.

Globular sets are not a rich enough invariant for homotopy, but maps of globular sets bear a criticall difference because of

$$\forall(\texttt{E:D}(\infty\texttt{-Grpd}_0)),\forall(\texttt{B:D}(\infty\texttt{-Grpd}_0)),\forall(\texttt{f:E} \longrightarrow \texttt{B}),\forall(\texttt{G:E} \longrightarrow \texttt{B}),(\forall(\texttt{n:Nat}),(\pi_n\ \texttt{F = }\pi_n\ \texttt{G})) \longrightarrow \texttt{F = G}$$

12. Globular Sets

The globe category $\mathbb{G}$ is the category

$$\begin{tikzcd} \bullet \arrow[r, shift left, "\sigma_0"] \arrow[r, shift right, "\tau_0"'] & \bullet \arrow[r, shift left, "\sigma_1"] \arrow[r, shift right, "\tau_1"'] & \bullet \arrow[r, shift left, "\sigma_2"] \arrow[r, shift right, "\tau_2"'] & \bullet \arrow[r, shift left, "\sigma_3"] \arrow[r, shift right, "\tau_3"'] & \bullet \arrow[r, shift left] \arrow[r, shift right] & \cdots \end{tikzcd}$$

Globular sets are functors from the opposite category of the globe category $\mathbb{G}$ into the category of sets, and maps of globular sets are natural transformations between them.

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.

Spheres and balls Next we turn to defining spheres and balls:

	Spheres and Balls	
Name of the X value	∂X $\cong$ S^n	X $\cong$ D^n
p-norm unit ball for p = 1	∂B(1,1)	B(1,1)
p-norm unit ball for 1 < p < 2	∂B(p,1)	B(p,1)
p-norm unit ball for p = 2	∂B(2,1)	B(2,1)
p-norm unit ball for 2 < p < ∞	∂B(p,1)	B(p,1)
p-norm unit ball for p = ∞	∂B(∞,1)	B(∞,1)
The n-simplex	$\partial\Delta^n$	Δ^n

Definition 1. ...

Theorem 1. i^1 : S^0 $\longrightarrow$ D^1

Theorem 2. D^n $\times$ D^1 $\longrightarrow$ D^{n+1}

Definition 2. $D^n \longrightarrow D^m$

Theorem 3. Fix n : $\mathbb{N}$ and let $\partial^n : S^n \longrightarrow D^{n+1}$ be the inclusion. The pushout of the following diagram is isomorphic to S^{n+1}:

$$\begin{tikzcd}
S^n \arrow[r] \arrow[d] & D^{n+1} \\
D^{n+1} &
\end{tikzcd}$$

Proof. □

Theorem 4. Define a function $||\text{-}||_2 : D^n \longrightarrow I$ sending $(x_1,\ldots,x_n)$ to $\sqrt{\sum_{i=1}^{n} x_i^2}$, and write $||\text{-}||_2$

Proof. ... □

13. HEP for based connected ∞-groupoids

In this section we prove the homotopy extension property for based connected ∞-groupoids, which we here model as CW-complexes.

Jar filling Next we turn to defining 'jar shapes' $\mathtt{J}^{\mathtt{n}}$, which include into $\mathtt{D}^{\mathtt{n}} \times \mathtt{I}$ $\mathtt{i}_n : \mathtt{J}^{\mathtt{n}} \longrightarrow \mathtt{D}^{\mathtt{n}} \times \mathtt{I}$, after which we 'fill' them (i.e. demonstrate that any continuous map $f : J^n \longrightarrow X$ extends to a continuous map $g : D^n \times I \longrightarrow X$).

The first and most common approach involves 'shining a light ray down from above the jar', i.e. projection. We obtain a formula for .

The second way to fill the jar

Change of Base Jar filling leaves the question

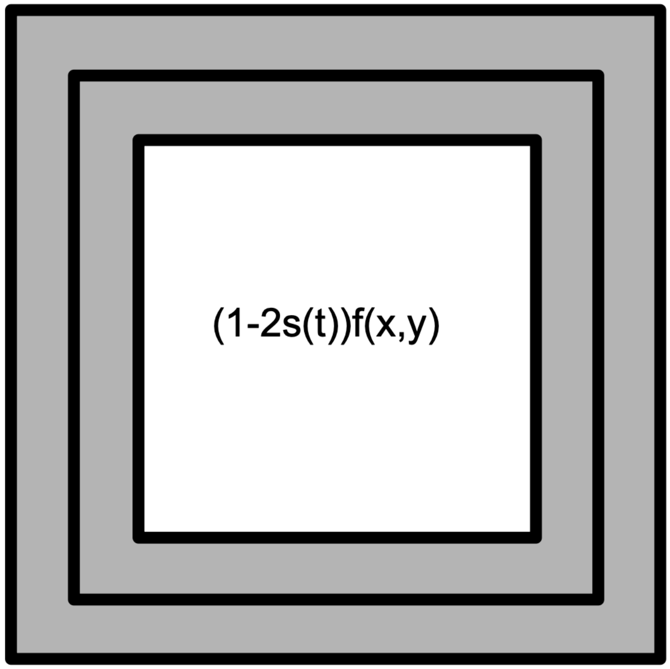
Definition 3. Let X_{-1} be a connected CW-complex and let n : Nat be a natural number. The transport function $\mathtt{trans\ n\ X}_{1} : (\mathtt{f} : [\mathtt{I},\mathtt{X}_{1}]) \to \pi_n\ (\mathtt{f\ 0}) \longrightarrow \pi_n\ (\mathtt{f\ 1})$ is

Theorem 5. Let X_{-1} be a connected CW-complex and let $f : I \longrightarrow X_{-1}$ be a path, so that $(\mathtt{trans\ n\ X}_{1}\ \mathtt{f}^{-1}) \bullet (\mathtt{trans\ n\ X}_{1}\ \mathtt{f})$ has type $\pi_n\ (\mathtt{f\ 0}) \longrightarrow \pi_n\ (\mathtt{f\ 0})$. Then

$$(\mathtt{trans\ n\ X}_{1}\ \mathtt{f}^{-1}) \bullet (\mathtt{trans\ n\ X}_{1}\ \mathtt{f}) = \mathtt{1\_}(\pi_n\ (\mathtt{f\ 0}))$$

Proof. ... □

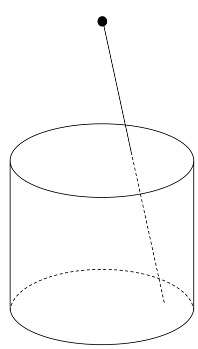
The proof in the above can be depicted like so, as a 'painting with two concentric frames':



that the based CW-complexes (X_{-1},x) and (X_{-1},y) are

Theorem 6.

Proof. ... □



14. REP for based connected ∞-groupoids

In this section we use the notion of globular sets to replace a topological space with a CW-complex. Together with HEP (homotopy extension), this will complete the proof of the Whitehead theorem.

15. The Whitehead theorem

Here we show the Whitehead theorem.

Chapter 3: The Category of Pairs

In this section I would like to

Chapter 4: Internal and Operadic Groups

16. InternalGroup

Lean 11

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

17. `InternalGroupAction 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).

18. OperadicGroup

19. OperadicGroupActions

20. The O and P Functors

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

21. The o and p Functors

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

PART 2: ∞-GROUPOIDS

The Whitehead theorem is about the ways that spheres can get stuck in spaces (higher homotopy groups), and the last section established how these higher homotopy groups relate to maps in the homotopy category of based connected CW-complexes. There is the nuance that the base .

Chapter 9: ∞-Grpd

22. 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.

23. ω

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

24. π_n

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

Chapter 10: The Whitehead Theorem for ∞-Groupoids

25. Cubical Complexes

…

1. Defining `repl`
2.

26. 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`

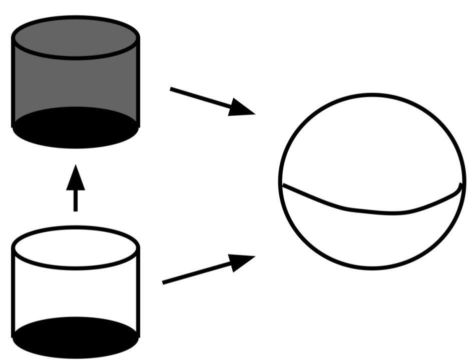
27. HEP

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.

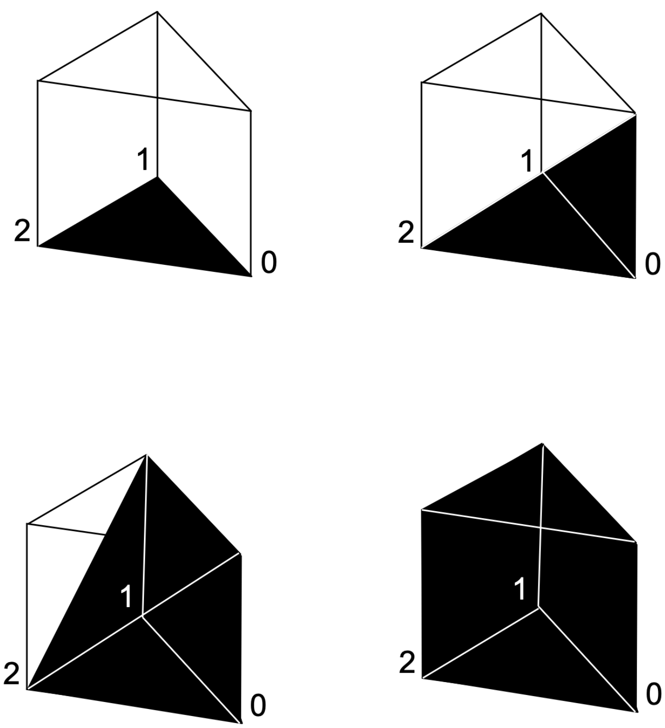


Prism Filling (PF) Let Y be a quasicategory, and let f, g : $\partial\Delta^n \longrightarrow Y$. A homotopy $h : \partial\Delta^n \times \Delta^1 \longrightarrow Y$ between f, g : $\partial\Delta^n \longrightarrow Y$ extends to a map $H : \Delta^n \times \Delta^1 \longrightarrow Y$;

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 $[\Delta^{n}, X]$ 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.

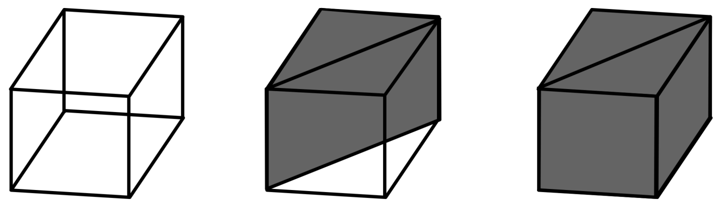
This means that we want directed box filling in addition to directed prism filling (but which also uses directed prism filling in its proof).

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: The Category of Pairs of ∞-Groupoids

Chapter 12: Internal and Operadic Groupoids and their 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(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 ($\vec{\text{P}}$_(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
`InternalGroupoid : Cat`	Internal categories
`InternalGroupoidAction G : 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
$\vec{\text{P}}$ C : InternalCategory D(∞-Cat)	$\vec{\Omega}$ C forms a component of an internal category
$\vec{\text{p}}$ (𝟙 C) D : InternalPresheaf D(∞-Cat/C) ($\vec{\text{P}}$ C)	$\vec{\omega}$ (𝟙 C) D forms a component of an internal C-presheaf

28. InternalGroupoid 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 `InternalGroupoid` in the way of composition.

Lean 12

```
-- definition of an internal category in a pullback
↪  system
/-
structure internal_groupoid (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 13

```
-- 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 14

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

Lean 15

```
-- definition of the composition of internal
↪  functors in a pullback system
def InternalCategoryCmp (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 16

```
-- proving the the first identity law for internal
↪  categories in a pullback system
def InternalCategoryId₁ (Γ : pullback_system) (X :
↪  internal_category Γ) (Y : internal_category Γ) (f
↪  : internal_functor Γ X Y) : InternalCategoryCmp Γ
↪  X Y Y f (InternalCategoryIdn Γ Y) = f := sorry
```

Lean 17

```
-- proving the second identity law for internal
↪  categories in a pullback system
def InternalCategoryId₂ (Γ : pullback_system) (X :
↪  internal_category Γ) (Y : internal_category Γ) (f
↪  : internal_functor Γ X Y) : (InternalCategoryCmp
↪  Γ X X Y (InternalCategoryIdn Γ X) f = f) := sorry
```

Lean 18

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

Lean 19

```
/-
def InternalCategory (Γ : pulback_system) :
↪  Cat.Obj := {Obj := internal_category Γ, Hom :=
↪  internal_functor Γ, Idn := InternalCategoryIdn
↪  Γ, Cmp := InternalCategoryCmp Γ, Id₁ :=
↪  InternalCategoryId₁ Γ, Id₂ :=
↪  InternalCategoryId₂ Γ, Ass :=
↪  InternalCategoryAss Γ}
-/
```

Lean 20

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

29. InternalGroupoidAction 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 21

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

Lean 22

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

Lean 23

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

Lean 24

```
-- defining the composition of internal functors
def Shfcmp (Γ : pullback_system) (C :
 ↪  (InternalCategory 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 25

```
-- proving the first identity law for internal
 ↪  functors
/-
def Shf... (Γ : pullback_system) (C :
 ↪  (InternalCategory 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 26

```
-- proving the second identity law for internal
 ↪  functors
/-
def ShfId₂ (Γ : pullback_system) (C :
 ↪  (InternalCategory 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 27

```
-- proving the associativity law for internal
↪  functors
/-
def ShfAss (Γ : pullback_system) (C :
↪  (InternalCategory 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 28

```
/-
def InternalPresheaf (Γ : pullback_system) (C :
↪  (InternalCategory 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 29

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

30. Operadic Groupoid

31. Operadic Groupoid Action

32. The P and O Functors

In this section we mention the internal category principal...

33. The p and o Functors

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.

PART 3: ∞-CATEGORIES

Chapter 13: ∞-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.

34. 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 $[\Delta^1,\text{-}] : \infty\text{-Cat} \longrightarrow \infty\text{-Cat}$. 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.

35. 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 $\vec{\mathrm{P}} : \mathrm{D}(\infty\text{-}\mathrm{Cat}) \longrightarrow \mathrm{InternalCategory}\ \mathrm{D}(\infty\text{-}\mathrm{Cat})$ and $\vec{\mathrm{p}} : \mathrm{D}(\infty\text{-}\mathrm{Cat}/\mathrm{C}) \longrightarrow \mathrm{InternalPresheaf}\ (\vec{\mathrm{P}}\ \mathrm{C})\ \mathrm{D}(\infty\text{-}\mathrm{Cat}/\mathrm{C})$.

36. π_n

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

Chapter 14: The Whitehead Theorem for ∞-Categories

37. Directed Cubical Complexes

...

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" 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 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 recognition theorems (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 $(\Delta^1)^{n-1} \longrightarrow \Delta^{1\,n}$, 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.

38. 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`

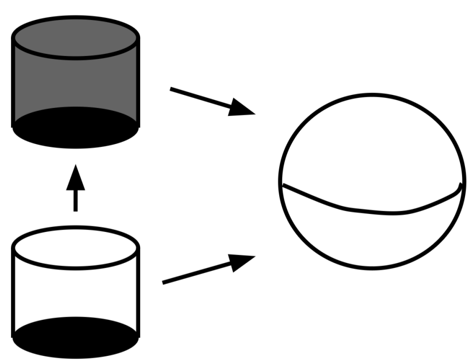
39. HEP

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.

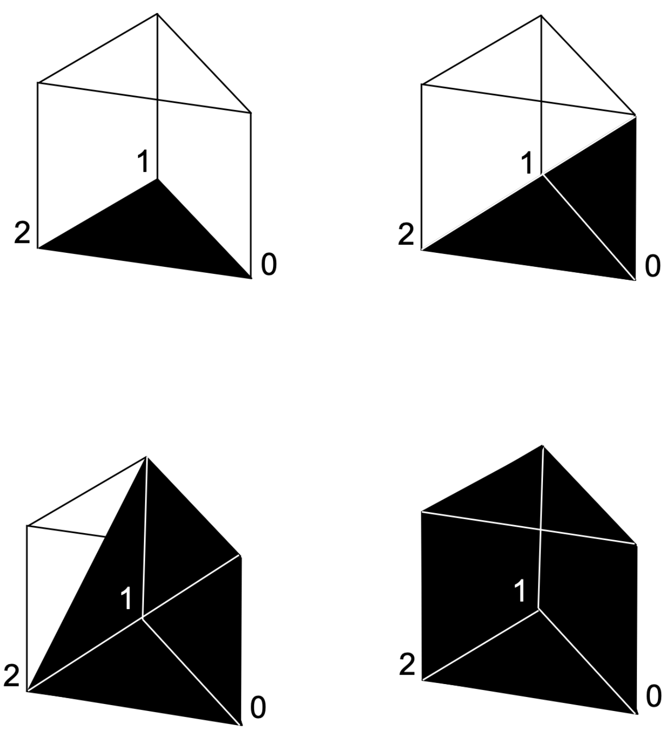


Prism Filling (PF) Let Y be a quasicategory, and let $f, g : \partial\Delta^n \longrightarrow Y$. A homotopy $h : \partial\Delta^n \times \Delta^1 \longrightarrow Y$ between $f, g : \partial\Delta^n \longrightarrow Y$ extends to a map $H : \Delta^n \times \Delta^1 \longrightarrow Y$;

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 $[\Delta^n, X]$ 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 $\vec{\Omega}^{\mathrm{n}}$ 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.

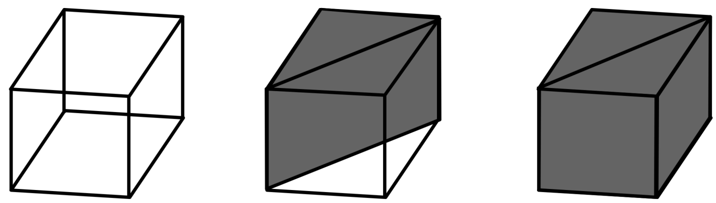
This means that we want directed box filling in addition to directed prism filling (but which also uses directed prism filling in its proof).

Box Filling (BF) Let Y be a quasicategory, and let f, g : $\partial\Delta^{\mathrm{n}} \longrightarrow$ Y. A homotopy h : $\partial\Delta^{\mathrm{n}} \times \Delta^1 \longrightarrow$ Y between f, g : $\partial\Delta^{\mathrm{n}} \longrightarrow$ 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 $\partial\Delta^{\mathrm{n}}$, 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 $\partial\Delta^{\mathrm{n}} \times \Delta^1$. 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 $\partial\Delta^{\mathrm{n}} \times \Delta^1$. 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 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:

40. 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 15: The Category of Pairs of ∞-Categories

…

Chapter 16: Internal and Operadic Categories and their 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 ($\vec{\text{P}}$_(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
`InternalCategory Γ : Cat`	Internal categories
`InternalPresheaf Γ C : Cat`	Internal C-presheaves
The internal category principal	`f ×_(B) f` forms an internal category
The internal presheaf principal	`f ×_(B)` g forms an internal presheaf
$\vec{\text{P}}$ C : InternalCategory D(∞-Cat)	$\vec{\Omega}$ C forms a component of an internal category
$\vec{\text{p}}$ (𝟙 C) D : InternalPresheaf D(∞-Cat/C) ($\vec{\text{P}}$ C)	$\vec{\omega}$ (𝟙 C) D forms a component of an internal C-presheaf

41. InternalCategory 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 `InternalCategory` in the way of composition.

Lean 30

```
-- 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 ...
-- Id_1 : D(Γ).
-- Id_2 : 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 31

```
-- 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 32

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

Lean 33

```
-- definition of the composition of internal
↪  functors in a pullback system
def InternalCategoryCmp (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 34

```
-- proving the the first identity law for internal
↪  categories in a pullback system
def InternalCategoryId₁ (Γ : pullback_system) (X :
↪  internal_category Γ) (Y : internal_category Γ) (f
↪  : internal_functor Γ X Y) : InternalCategoryCmp Γ
↪  X Y Y f (InternalCategoryIdn Γ Y) = f := sorry
```

Lean 35

```
-- proving the second identity law for internal
↪ categories in a pullback system
def InternalCategoryId₂ (Γ : pullback_system) (X :
↪ internal_category Γ) (Y : internal_category Γ) (f
↪ : internal_functor Γ X Y) : (InternalCategoryCmp
↪ Γ X X Y (InternalCategoryIdn Γ X) f = f) := sorry
```

Lean 36

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

Lean 37

```
/-
def InternalCategory (Γ : pulback_system) :
↪ Cat.Obj := {Obj := internal_category Γ, Hom :=
↪ internal_functor Γ, Idn := InternalCategoryIdn
↪ Γ, Cmp := InternalCategoryCmp Γ, Id₁ :=
↪ InternalCategoryId₁ Γ, Id₂ :=
↪ InternalCategoryId₂ Γ, Ass :=
↪ InternalCategoryAss Γ}
-/
```

Lean 38

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

42. InternalPresheaf 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 39

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

Lean 40

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

Lean 41

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

Lean 42

```
-- defining the composition of internal functors
def Shfcmp (Γ : pullback_system) (C :
 ↪  (InternalCategory 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 43

```
-- proving the first identity law for internal
 ↪  functors
/-
def Shf... (Γ : pullback_system) (C :
 ↪  (InternalCategory 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 44

```
-- proving the second identity law for internal
 ↪  functors
/-
def ShfId₂ (Γ : pullback_system) (C :
 ↪  (InternalCategory 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 45

```
-- proving the associativity law for internal
↪  functors
/-
def ShfAss (Γ : pullback_system) (C :
↪  (InternalCategory 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 46

```
/-
def InternalPresheaf (Γ : pullback_system) (C :
↪  (InternalCategory 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 47

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

43. The P and O Functors

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.

44. The p and o Functors

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.

Bibliography

1. Davis, James F., and Paul Kirk. Lecture notes in algebraic topology. Vol. 35. Providence: American Mathematical Society, 2001.

2.

3. Galois theory and a general notion of central simple extension (Janelidze)

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

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

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

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

8. 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.

9. 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.

10. 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.

11. `https://www.uni-muenster.de/IVV5WS/WebHop/user/nikolaus/Papers/oo-bundles_general_theory.pdf`

12. `https://www.cse.chalmers.se/~coquand/cubicaltt.pdf`

13. `https://arxiv.org/pdf/1607.04156.pdf`

14. `https://carloangiuli.com`

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.
8. A comment thread concerning Jacob Lurie’s breakthrough prize and different approaches to homotopy on the computer
9. Arlin, Kevin David. ”2-categorical Brown representability and the relation between derivators and infinity-categories.” Doctoral dissertation, University of California, Los Angeles, 2020.
10.

Some 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`

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.