A Formalization of the Algebraic Small Object Argument in `UniMath`

Dennis Hilhorst

Supervisor
Dr. Paige North

Second reader
Dr. Jaap van Oosten

12-12-2023

Utrecht University

Abstract

Model categories, introduced by Quillen in 1967, form the cornerstone of modern homotopy theory, providing a language and tools for this branch of mathematics. They consist of two interacting weak factorization systems. Quillen defined a transfinite construction to generate weak factorization systems and thereby model structures on a category, given sufficiently well-behaved classes of maps: the *small object argument*. Weak factorization systems, lacking algebraic structure, suffer some defects from a categorical point of view. Grandis and Tholen introduced the notion of *natural weak factorization system* to rectify these issues. Garner pointed out some problematic aspects of the small object argument: that it is not convergent, that it is not related to other known transfinite constructions and that it satisfies no universal property. He refined the small object argument to generate natural weak factorization systems in a more algebraically coherent way.

In this thesis, we elaborate, rephrase and formalize Garner's 'algebraic' small object argument. The formalization is written using the Coq proof checker, using the `UniMath` library. This is a formalization framework based on Homotopy Type Theory (HoTT). The formalization provides an air-tight confirmation of the theory through computer verified proofs.

We fill in the details in Garner's construction, add much needed intuition and redefine parts of the construction to be more direct and accessible. We rephrase the theory in more modern language, using constructions like *displayed categories* and a modern, less strict notion of *monoidal categories*, so that it is fit for formalization. We point out the interaction between the theory and the HoTT foundations, and describe some of the constructive issues we come across.

Abstract

Model categories, introduced by Quillen in 1967, form the cornerstone of modern homotopy theory, providing a language and tools for this branch of mathematics. They consist of two interacting weak factorization systems. Quillen defined a transfinite construction to generate weak factorization systems and thereby model structures on a category, given sufficiently well-behaved classes of maps: the *small object argument*. Weak factorization systems, lacking algebraic structure, suffer some defects from a categorical point of view. Grandis and Tholen introduced the notion of *natural weak factorization system* to rectify these issues. Garner pointed out some problematic aspects of the small object argument: that it is not convergent, that it is not related to other known transfinite constructions and that it satisfies no universal property. He refined the small object argument to generate natural weak factorization systems in a more algebraically coherent way.

In this thesis, we elaborate, rephrase and formalize Garner's 'algebraic' small object argument. The formalization is written using the Coq proof checker, using the `UniMath` library. This is a formalization framework based on Homotopy Type Theory (HoTT). The formalization provides an air-tight confirmation of the theory through computer verified proofs.

We fill in the details in Garner's construction, add much needed intuition and redefine parts of the construction to be more direct and accessible. We rephrase the theory in more modern language, using constructions like *displayed categories* and a modern, less strict notion of *monoidal categories*, so that it is fit for formalization. We point out the interaction between the theory and the HoTT foundations, and describe some of the constructive issues we come across.

Abstract

Model categories, introduced by Quillen in 1967, form the cornerstone of modern homotopy theory, providing a language and tools for this branch of mathematics. They consist of two interacting weak factorization systems. Quillen defined a transfinite construction to generate weak factorization systems and thereby model structures on a category, given sufficiently well-behaved classes of maps: the *small object argument*. Weak factorization systems, lacking algebraic structure, suffer some defects from a categorical point of view. Grandis and Tholen introduced the notion of *natural weak factorization system* to rectify these issues. Garner pointed out some problematic aspects of the small object argument: that it is not convergent, that it is not related to other known transfinite constructions and that it satisfies no universal property. He refined the small object argument to generate natural weak factorization systems in a more algebraically coherent way.

In this thesis, we elaborate, rephrase and formalize Garner's 'algebraic' small object argument. The formalization is written using the Coq proof checker, using the `UniMath` library. This is a formalization framework based on Homotopy Type Theory (HoTT). The formalization provides an air-tight confirmation of the theory through computer verified proofs.

We fill in the details in Garner's construction, add much needed intuition and redefine parts of the construction to be more direct and accessible. We rephrase the theory in more modern language, using constructions like *displayed categories* and a modern, less strict notion of *monoidal categories*, so that it is fit for formalization. We point out the interaction between the theory and the HoTT foundations, and describe some of the constructive issues we come across.

Abstract

Model categories, introduced by Quillen in 1967, form the cornerstone of modern homotopy theory, providing a language and tools for this branch of mathematics. They consist of two interacting weak factorization systems. Quillen defined a transfinite construction to generate weak factorization systems and thereby model structures on a category, given sufficiently well-behaved classes of maps: the *small object argument*. Weak factorization systems, lacking algebraic structure, suffer some defects from a categorical point of view. Grandis and Tholen introduced the notion of *natural weak factorization system* to rectify these issues. Garner pointed out some problematic aspects of the small object argument: that it is not convergent, that it is not related to other known transfinite constructions and that it satisfies no universal property. He refined the small object argument to generate natural weak factorization systems in a more algebraically coherent way.

In this thesis, we elaborate, rephrase and formalize Garner's 'algebraic' small object argument. The formalization is written using the Coq proof checker, using the `UniMath` library. This is a formalization framework based on Homotopy Type Theory (HoTT). The formalization provides an air-tight confirmation of the theory through computer verified proofs.

We fill in the details in Garner's construction, add much needed intuition and redefine parts of the construction to be more direct and accessible. We rephrase the theory in more modern language, using constructions like *displayed categories* and a modern, less strict notion of *monoidal categories*, so that it is fit for formalization. We point out the interaction between the theory and the HoTT foundations, and describe some of the constructive issues we come across.

Motivation

Model Categories:

- Language and tools for doing math ‘up to homotopy’.
- Applications in many fields of mathematics
  - Homotopy theory
  - Algebraic geometry
  - Condensed mathematics
  - Computer science

(Algebraic) Small Object Argument:

- Generate (N)WFSs and thereby (algebraic) model structures.
- Generation from simple, small data.
- (Algebraically coherent and convergent construction.)

Contributions:

- Formalization of the theory.
- Detail, intuition and modifications.
- Interactions between theory and HoTT.
- Reformulation of theory using modern constructions.

Motivation

Model Categories:

- Language and tools for doing math ‘up to homotopy’.
- Applications in many fields of mathematics
  - Homotopy theory
  - Algebraic geometry
  - Condensed mathematics
  - Computer science

(Algebraic) Small Object Argument:

- Generate (N)WFSs and thereby (algebraic) model structures.
- Generation from simple, small data.
- (Algebraically coherent and convergent construction.)

Contributions:

- Formalization of the theory.
- Detail, intuition and modifications.
- Interactions between theory and HoTT.
- Reformulation of theory using modern constructions.

Motivation

Model Categories:

- Language and tools for doing math ‘up to homotopy’.
- Applications in many fields of mathematics
  - Homotopy theory
  - Algebraic geometry
  - Condensed mathematics
  - Computer science

(Algebraic) Small Object Argument:

- Generate (N)WFSs and thereby (algebraic) model structures.
- Generation from simple, small data.
- (Algebraically coherent and convergent construction.)

Contributions:

- Formalization of the theory.
- Detail, intuition and modifications.
- Interactions between theory and HoTT.
- Reformulation of theory using modern constructions.

Outline

Outline

What is Formalization?

Generally, people tend to speak of formal arguments when referring to arguments by formalism, hence by purely symbolic manipulations following syntactic rewrite rules.

(Some nLab user)

From `https://ncatlab.org/nlab/show/proof`.

A Short Example

```
Lemma eq_symm (x y : nat) :
    x = y -> y = x.
Proof.
  intro H.
  rewrite H.
  reflexivity.
Qed.
```

```
x, y : nat
H : x = y
```

Goal:

```
x = y -> y = x
```

```
(* try it yourself at
   nng.dennishilhorst.nl *)
```

A Short Example

```
Lemma eq_symm (x y : nat) :
    x = y -> y = x.
Proof.
  intro H.
  rewrite H.
  reflexivity.
Qed.
```

```
(* try it yourself at
   nng.dennishilhorst.nl *)
```

```
x, y : nat
H : x = y
```

Goal:

```
y = x
```

A Short Example

```
Lemma eq_symm (x y : nat) :
    x = y -> y = x.
Proof.
  intro H.
  rewrite H.
  reflexivity.
Qed.
```

```
(* try it yourself at
   nng.dennishilhorst.nl *)
```

```
x, y : nat
H : x = y
```

Goal:

```
y = y
```

A Short Example

```
Lemma eq_symm (x y : nat) :
    x = y -> y = x.
Proof.
  intro H.
  rewrite H.
  reflexivity.
Qed.
```

```
(* try it yourself at
   nng.dennishilhorst.nl *)
```

```
x, y : nat
H : x = y
```

Goal:

Proof finished

Another Short Example

```
Context {C : category} (CC : Colims C).
Context {g : graph} (d : diagram g (LNWFStot C)).
Context (H : is_connected g) (v0 : vertex g).

Lemma ColimLNWFSCocone : ColimCocone d.
Proof.
  use make_ColimCocone.
  - exists (colim Finf).
    exact (ColimLNWFS_disp).
  - use make_cocone.
    * intro v.
      exists (colimIn Finf v).
      exact (ColimLNWFS_disp_in v).
    * abstract (
        intros u v e;
        apply subtypePath;
            [intro; apply isaprop_lnwfs_mor_axioms|];
        exact (colimInCommutes Finf _ _ e)
      ).
  - intros L cc; exact (ColimLNWFS_unique cc).
Defined.
```

Outline

Set Theory vs Type Theory

We note that a set-theoretic foundation has two 'layers': the deductive system of first-order logic, and, formulated inside this system, the axioms of a particular theory, such as ZFC.

...

By contrast, type theory is its own deductive system: it need not be formulated inside any superstructure, such as first-order logic. (The HoTT Book)

Set Theory vs Type Theory

We note that a set-theoretic foundation has two 'layers': the deductive system of first-order logic, and, formulated inside this system, the axioms of a particular theory, such as ZFC.

...

By contrast, type theory is its own deductive system: it need not be formulated inside any superstructure, such as first-order logic. (The HoTT Book)

$\Longrightarrow$ In type theory: *everything* is a type!

Set Theory vs Type Theory

Set theory	**Type theory**
set A	type A
member $a \in A$	term $a : A$

Set Theory vs Type Theory

Set theory	**Type theory**
set A	type A
member $a \in A$	term $a : A$

$a : A$ is a *judgement*, $a \in A$ is a *proposition*.

Set Theory vs Type Theory

Set theory	**Type theory**
set A	type A
member $a \in A$	term $a : A$
proposition P	type P

$a : A$ is a *judgement*, $a \in A$ is a *proposition*.

Set Theory vs Type Theory

Set theory	**Type theory**
set A	type A
member $a \in A$	term $a : A$
proposition P	type P
'P has a proof'	term $p : P$

$a : A$ is a *judgement*, $a \in A$ is a *proposition*.

Set Theory vs Type Theory

Set theory	**Type theory**
set A	type A
member $a \in A$	term $a : A$
proposition P	type P
'P has a proof'	term $p : P$
proposition $a = b$	type $a =_A b$

$a : A$ is a *judgement*, $a \in A$ is a *proposition*.

Set Theory vs Type Theory

Set theory	**Type theory**
set A	type A
member $a \in A$	term $a : A$
proposition P	type P
'P has a proof'	term $p : P$
proposition $a = b$	type $a =_A b$
'proof of $a = b$'	term $q : (a =_A b)$

$a : A$ is a *judgement*, $a \in A$ is a *proposition*.

Set Theory vs Type Theory

Set theory	**Type theory**
set A	type A
member $a \in A$	term $a : A$
proposition P	type P
'P has a proof'	term $p : P$
proposition $a = b$	type $a =_A b$
'proof of $a = b$'	term $q : (a =_A b)$
definition $a = b$	definitional equality $a := b$

$a : A$ is a *judgement*, $a \in A$ is a *proposition*.

Set Theory vs Type Theory

Set theory	**Type theory**
set A	type A
member $a \in A$	term $a : A$
proposition P	type P
'P has a proof'	term $p : P$
proposition $a = b$	type $a =_A b$
'proof of $a = b$'	term $q : (a =_A b)$
definition $a = b$	definitional equality $a := b$

$a : A$ is a *judgement*, $a \in A$ is a *proposition*.
$a := b$ is a *judgement*, $a =_A b$ is a *proposition*.

Definitional vs Propositional Equality

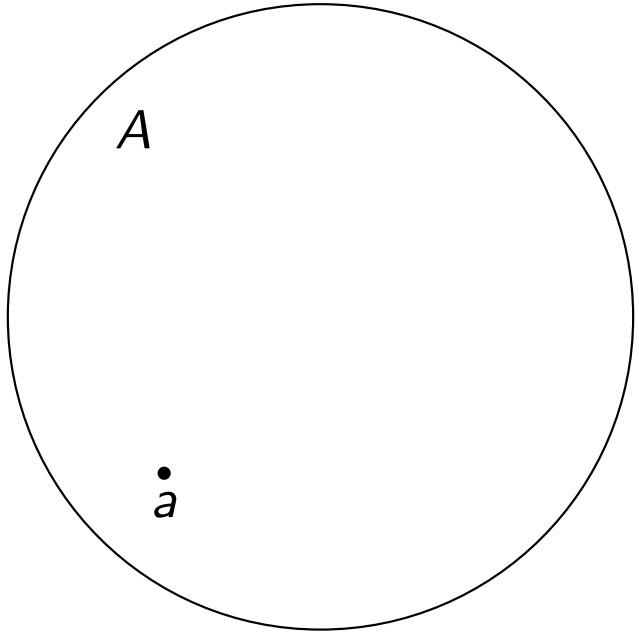
```
Lemma eq_symm
    (x y : nat)
    (H : x = y) :
  y = x.
Proof.
  rewrite H.
  reflexivity.
Qed.
```

```
Lemma looks_like_eq_symm
    (y : nat)
    (x := y) :
  y = x.
Proof.

  reflexivity.
Qed.
```

We like definitional equalities better!

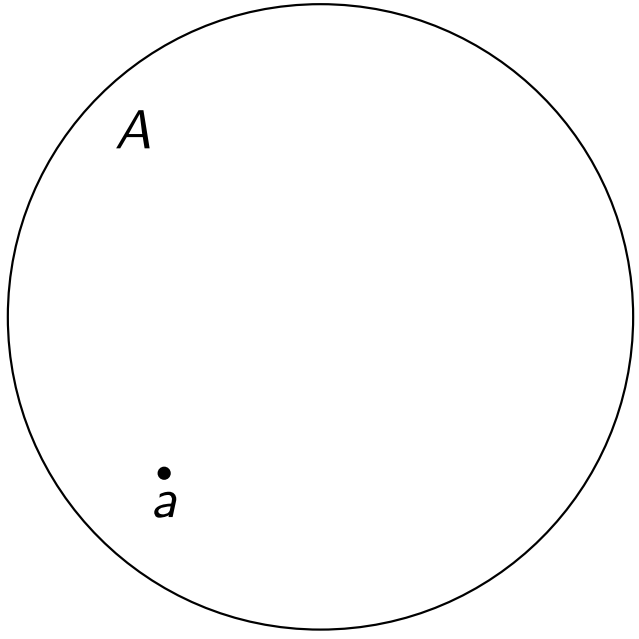
Homotopy Type Theory



$a : A$

```
a : A
```

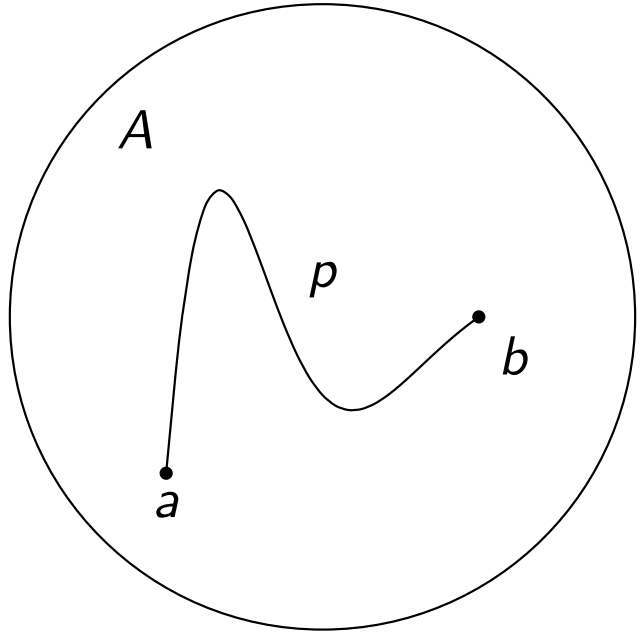
Homotopy Type Theory



$$\mathsf{refl}_a : a =_A a$$

```
reflexivity.
```

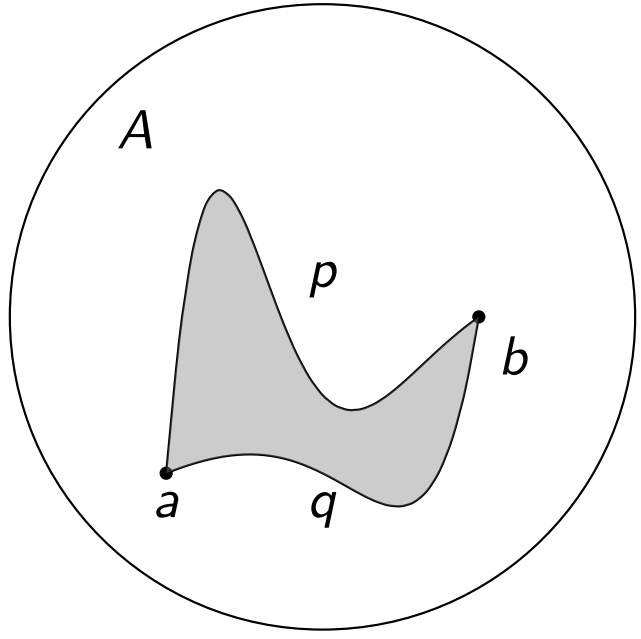
Homotopy Type Theory



$$p : a =_A b$$

```
rewrite p.
```

Homotopy Type Theory



$$H : p =_{a=_A b} q$$

```
rewrite H.
```

Interpretations

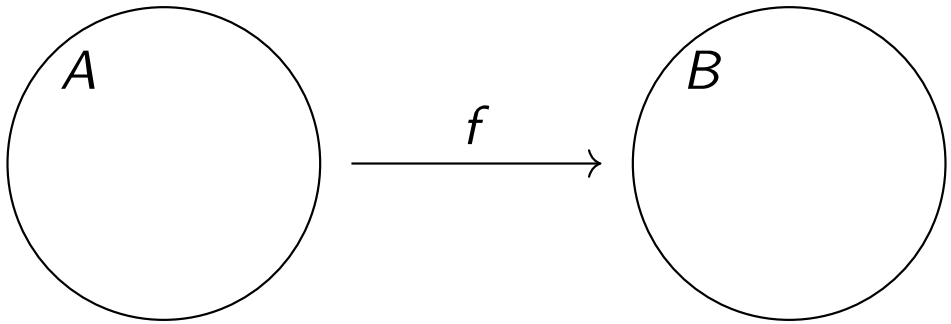
Types	Sets	Logic	Homotopy
A	set	proposition	space
$a : A$	element	proof	point
`unit`,`empty`	$\{ \emptyset \}$, $\emptyset$	true, false	$*$, $\emptyset$
$a =_A b$	$\{ (a, a) \mid a \in A \}$	equality	path space

Outline

Function Types

Types $f : A \to B$

Homotopy



Sets $f : A \to B$

Logic $f : A \implies B$

Type Universes

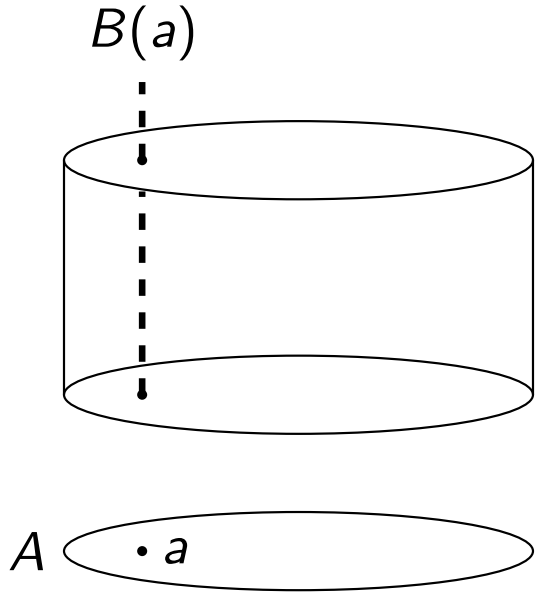
$$A : \mathcal{U} \implies \text{`}A \text{ is a type'}$$

$\mathcal{U}$ (`UU` in `UniMath`) is a 'large enough universe of types'

Dependent Types

Types $B : A \to \mathcal{U}$

Homotopy



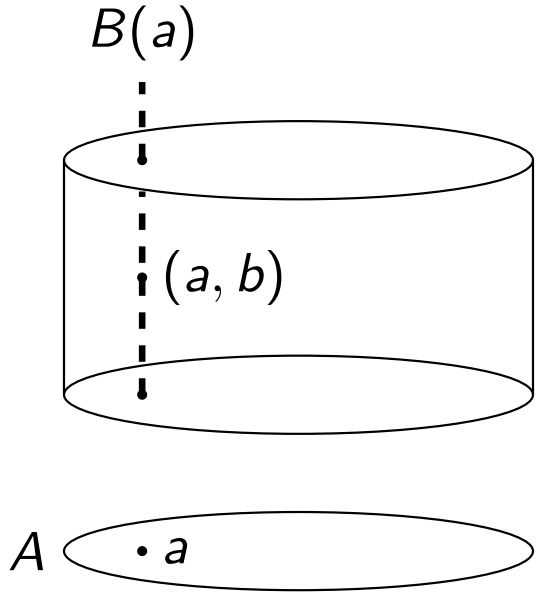
Sets $\{ B_a \}_{a \in A}$

Logic $B(a)$ for $a : A$

Dependent Types

Types $\sum_{a:A} B(a)$

Homotopy



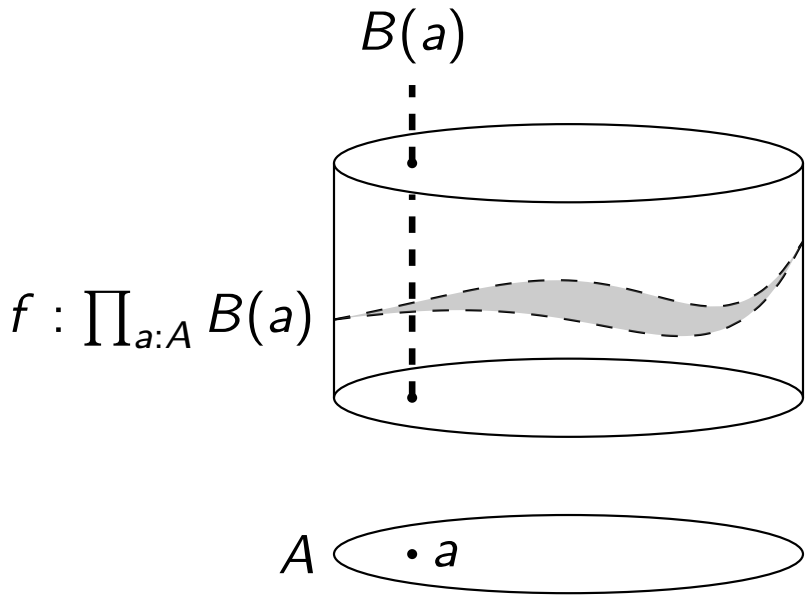
Sets $\sum_{a \in A} B_a$

Logic $\exists_{a:A} B(a)$

Dependent Types

Types $\prod_{a:A} B(a)$

Homotopy



Sets $\prod_{a \in A} B_a$

Logic $\forall_{a:A} B(a)$

$\prod$-types

```
Lemma eq_symm (x y : nat) : x = y -> y = x.
```

$$\texttt{eq\_symm} : \prod_{x:\mathbb{N}} \left(\prod_{y:\mathbb{N}} x =_{\mathbb{N}} y \to y =_{\mathbb{N}} x \right)$$

```
λ (x y : nat) (H : x = y),
    match H in (_ = y0) return (y = y0 → y = x) with
    | idpath _ => λ HC : y = x, HC
    end (idpath y)
```

$\sum$-types

- Type of terms with property

$$\texttt{odd\_numbers} : \sum_{x:\mathbb{N}} \texttt{is\_odd}\ x.$$

- Dependent data

$$\texttt{pointed\_space} : \sum_{X:\mathbf{TOP}} X.$$

Example

The product type $A \times B$ is a $\sum$-type with a constant type family

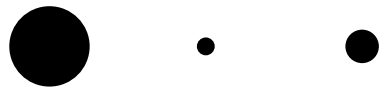
$$\sum_{a:A} B.$$

Interpretations

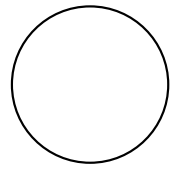
Types	Sets	Logic	Homotopy
A	set	proposition	space
$a : A$	element	proof	point
$B : A \to \mathcal{U}$	family of sets	predicate	fibration
`unit`,`empty`	$\{ \emptyset \}$, $\emptyset$	true, false	$*$, $\emptyset$
$A \times B$	set of pairs	$A \wedge B$	product space
$A \to B$	set of functions	$A \implies B$	function space
$\sum_{a:A} B(a)$	disjoint sum	$\exists(a : A), B(a)$	total space
$\prod_{a:A} B(a)$	product	$\forall(a : A), B(a)$	space of sections
$a =_A b$	$\{ (a, a) \mid a \in A \}$	equality	path space

n-types

0-type



1-type



2-type

...

n-types

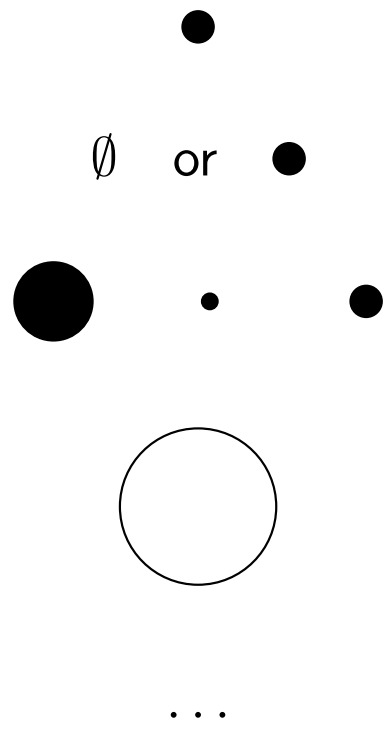
(-2)-type

(-1)-type

0-type

1-type

2-type



Mere Propositions

$$\texttt{hProp} := \sum_{P:\mathcal{U}} \texttt{isaprop}\ P$$

- Behave like propositions in 'classical' logic.
- Sometimes we *need* a type to be propositional:

```
hsubtype (A : UU) := A -> hProp.
```

- Save *opaque* proof terms.

```
λ (x y : nat) (H : x = y),
  match H in (_ = y0) return (y = y0 →
   ↪  y = x) with
  | idpath _ => λ HC : y = x, HC
  end (idpath y)
```

} eq_symm

Mere Propositions

$$\texttt{hProp} := \sum_{P:\mathcal{U}} \texttt{isaprop}\ P$$

- Behave like propositions in 'classical' logic.
- Sometimes we *need* a type to be propositional:

```
hsubtype (A : UU) := A -> hProp.
```

- Save *opaque* proof terms.

```
λ (x y : nat) (H : x = y),
  match H in (_ = y0) return (y = y0 →
   ↪  y = x) with
  | idpath _ => λ HC : y = x, HC
  end (idpath y)
```

} eq_symm

Mere Propositions

$$\texttt{hProp} := \sum_{P:\mathcal{U}} \texttt{isaprop}\ P$$

- Behave like propositions in 'classical' logic.
- Sometimes we *need* a type to be propositional:

```
hsubtype (A : UU) := A -> hProp.
```

- Save *opaque* proof terms.

```
λ (x y : nat) (H : x = y),
   match H in (_ = y0) return (y = y0 →
   ↪  y = x) with
   | idpath _ => λ HC : y = x, HC
   end (idpath y)
```

} eq_symm

Propositional Truncation

Definition

Let $A : \mathcal{U}$, there is a type $\|A\|$ called the *propositional truncation* of A, with the universal property

$$\prod_{P:\mathtt{hProp}} (A \to P) \to (\|A\| \to P).$$

- $\|A\|$ is inhabited whenever A is.
- $\|A\|$ is propositional.
- **A term $a : \|A\|$ does not give a term of A.**

Outline

Precategories

Set theory	category	Set of objects $\mathcal{C}_0$ Sets of morphisms $\mathrm{Hom}(X, Y)$
Type theory	category?	Type of objects $\mathcal{C}_0$ Types of morphisms $\mathrm{Hom}(X, Y)$

Precategories

Set theory	category	Set of objects $\mathcal{C}_0$ Sets of morphisms $\mathrm{Hom}(X, Y)$
Type theory	precategory	Type of objects $\mathcal{C}_0$ Types of morphisms $\mathrm{Hom}(X, Y)$

Precategories

Set theory	category	Set of objects $\mathcal{C}_0$ Sets of morphisms $\mathrm{Hom}(X, Y)$
Type theory	precategory	Type of objects $\mathcal{C}_0$ Types of morphisms $\mathrm{Hom}(X, Y)$
Type theory	category	Type of objects $\mathcal{C}_0$ Sets of morphisms $\mathrm{Hom}(X, Y)$

Precategories

Set theory	category	Set of objects $\mathcal{C}_0$ Sets of morphisms $\mathrm{Hom}(X, Y)$
Type theory	precategory	Type of objects $\mathcal{C}_0$ Types of morphisms $\mathrm{Hom}(X, Y)$
Type theory	category	Type of objects $\mathcal{C}_0$ Sets of morphisms $\mathrm{Hom}(X, Y)$
Type theory	setcategory	Set of objects $\mathcal{C}_0$ Sets of morphisms $\mathrm{Hom}(X, Y)$

Precategories

Set theory	category	Set of objects $\mathcal{C}_0$ Sets of morphisms $\mathrm{Hom}(X, Y)$
Type theory	precategory	Type of objects $\mathcal{C}_0$ Types of morphisms $\mathrm{Hom}(X, Y)$
Type theory	category	Type of objects $\mathcal{C}_0$ Sets of morphisms $\mathrm{Hom}(X, Y)$
Type theory	setcategory	Set of objects $\mathcal{C}_0$ Sets of morphisms $\mathrm{Hom}(X, Y)$
Type theory	univalent category	Type of objects $\mathcal{C}_0$ Sets of morphisms $\mathrm{Hom}(X, Y)$ Equality $\iff$ Isomorphism

$$\texttt{idtoiso} : (X =_{\mathcal{C}} Y) \to (\texttt{z\_iso}\ X\ Y)$$

Morphism Classes

```
Definition morphism_class (C : category) : UU :=
    ∏ (X Y : C), hsubtype (X --> Y).
```

Example

The morphism class containing all morphisms of $\mathcal{C}$, we will call this `univ`.

Example

The morphism class containing all isomorphisms of $\mathcal{C}$, we will call this `isos`.

Lifting Problems

Definition

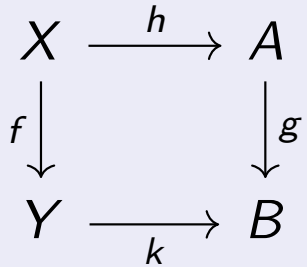
For two morphisms f, g in $\mathcal{C}$, an (f, g)-*lifting problem* is a diagram

$$\begin{array}{ccc} X & \xrightarrow{h} & A \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ Y & \xrightarrow[k]{} & B \end{array}$$

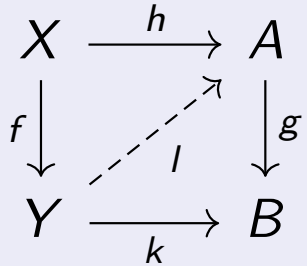
Lifting Problems

Definition

For two morphisms f, g in $\mathcal{C}$, an (f, g)-*lifting problem* is a diagram



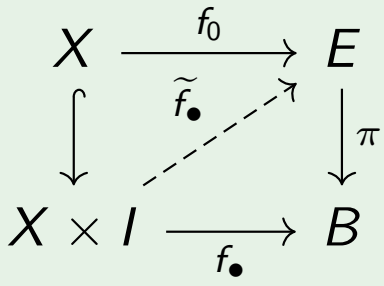
The lifting problem has a filler l if the dashed arrow exists.



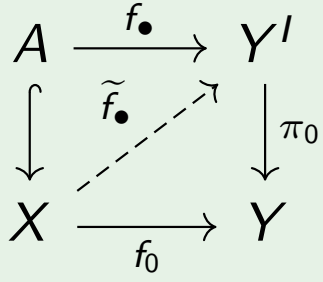

```
Definition filler {x y a b : C} {f : x --> y} {g : a --> b}
        {h : x --> a} {k : y --> b} (H : h · g = f · k) :=
    ∑ l : y --> a, (f · l = h) × (l · g = k).
```

Lifting Problems

Example (The Homotopy Lifting Property)



Example (The Homotopy Extension Property)



Lifting Property

Definition

For a class of maps $\mathcal{L}$, we define $\mathcal{L}^{\square}$ the class of all maps g for which any

$$\begin{array}{ccc} X & \longrightarrow & A \\ {\scriptstyle f \in \mathcal{L}}\downarrow & & \downarrow{\scriptstyle g} \\ Y & \longrightarrow & B \end{array}$$

has a (not necessarily unique) filler.

```
Definition lp {x y a b : C}
      (f : x --> y) (g : a --> b) : hProp :=
    ∀ (h : x --> a) (k : y --> b) (H : h · g = f · k),
      ∥filler H∥.

Definition rlp (L : morphism_class C) : (morphism_class C) :=
    λ {a b : C} (g : a --> b),
      ∀ (x y : C) (f : x --> y),
        ((L _ _) f ⇒ lp f g).
```

Lifting Property

Definition

For a class of maps $\mathcal{R}$, we define ${}^{\square}\mathcal{R}$ the class of all maps f for which any

$$\begin{array}{ccc} X & \longrightarrow & A \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g \in \mathcal{R}} \\ Y & \longrightarrow & B \end{array}$$

has a (not necessarily unique) filler.

```
Definition lp {x y a b : C}
      (f : x --> y) (g : a --> b) : hProp :=
    ∀ (h : x --> a) (k : y --> b) (H : h · g = f · k),
      ∥filler H∥.

Definition llp (R : morphism_class C) : (morphism_class C) :=
    λ {x y : C} (f : x --> y),
      ∀ (a b : C) (g : a --> b),
        ((R _ _) g ⇒ lp f g).
```

Lifting Property

Example

$\mathtt{isos}^{\square} = \mathtt{univ}$, as well as ${}^{\square}\mathtt{isos} = \mathtt{univ}$. Similarly, $\mathtt{univ}^{\square} = \mathtt{isos}$, as well as ${}^{\square}\mathtt{univ} = \mathtt{isos}$

Example

The HLP is used to define fibrations as right class of $\{ X \hookrightarrow X \times I \}$.

Example

The HEP is used to define cofibrations as left class of $\{ Y^I \to Y \}$.

Factorizations

Definition

We say that a pair of morphism classes $(\mathcal{L}, \mathcal{R})$ *factors* $\mathcal{C}$ if every morphism $f : X \to Y$ factors as a composite

$$X \xrightarrow{\lambda} E_f \xrightarrow{\rho} Y$$

with $E_f : \mathcal{C}$, $\lambda \in \mathcal{L}$ and $\rho \in \mathcal{R}$.

```
Definition wfs_fact_ax {C : category} (L R : morphism_class C) :=
  ∏ x y (f : x --> y),
    ‖∑ ef (l : x --> ef) (r : ef --> y),
      (L _ _) l × (R _ _) r × l · r = f‖.
```

Example

(`isos`, `univ`) and (`univ`, `isos`) factor any category.

Weak Factorization Systems

Definition

A weak factorization system (WFS), is an ordered pair $(\mathcal{L}, \mathcal{R})$ of classes of maps in $\mathcal{C}$ that factors $\mathcal{C}$ and satisfies

$$\mathcal{L} = {}^{\square}\mathcal{R} \quad \text{and} \quad \mathcal{R} = \mathcal{L}^{\square}.$$

```
Definition is_wfs {C : category} (L R : morphism_class C) :=
  (L = llp R) × (R = rlp L) × (wfs_fact_ax L R).

Definition wfs (C : category) :=
  ∑ (L R : morphism_class C), is_wfs L R.
```

Weak Factorization Systems

Example

$(\mathtt{isos}, \mathtt{univ})$ and $(\mathtt{univ}, \mathtt{isos})$ define WFSs on any category.

Example

The HLP defines a WFS on **TOP**.

Example

The HEP defines a WFS on **TOP**.

Properties of WFSs

Lemma

Let $(\mathcal{L}, \mathcal{R})$ be a WFS in a category $\mathcal{C}$. Then the following properties hold.

1. *$\mathcal{L}$ contains all isomorphisms of $\mathcal{C}$.*
2. *$\mathcal{L}$ is closed under retracts.*
3. *Any pushout of a map in $\mathcal{L}$ is in $\mathcal{L}$. That is, if the following diagram is a pushout and f is in $\mathcal{L}$, then so is p_1.*

$$\begin{array}{ccc} X & \longrightarrow & Z \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle p_1} \\ Y & \longrightarrow & P \end{array}$$

4. *Any coproduct of maps in $\mathcal{L}$ is in $\mathcal{L}$.*
5. *$\mathcal{L}$ is closed under transfinite composition. That is, for a chain of $\mathcal{L}$-maps*

$$X_0 \to X_1 \to X_2 \to \dots,$$

the composite $X_0 \to \operatorname{colim} X_v$ is in $\mathcal{L}$.

Properties of WFSs

Lemma

Let $(\mathcal{L}, \mathcal{R})$ be a WFS in a category $\mathcal{C}$. Then the following properties hold.

1. *$\mathcal{L}$ contains all isomorphisms of $\mathcal{C}$.*
2. *$\mathcal{L}$ is closed under retracts.*
3. *Any pushout of a map in $\mathcal{L}$ is in $\mathcal{L}$. That is, if the following diagram is a pushout and f is in $\mathcal{L}$, then so is p_1.*

$$\begin{array}{ccc} X & \longrightarrow & Z \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle p_1} \\ Y & \longrightarrow & P \end{array}$$

4. *Any coproduct of maps in $\mathcal{L}$ is in $\mathcal{L}$.*
5. *$\mathcal{L}$ is closed under transfinite composition. That is, for a chain of $\mathcal{L}$-maps*

$$X_0 \to X_1 \to X_2 \to \dots,$$

the composite $X_0 \to \operatorname{colim} X_v$ is in $\mathcal{L}$.

Closure Under Coproducts

Suppose all $f_i \in \mathcal{L}$ for $i : I$, is $\bigsqcup_{i:I} f_i \in \mathcal{L}$?

$$\begin{array}{ccc} \bigsqcup_{i:I} X_i & \longrightarrow & A \\ {\scriptstyle \bigsqcup_{i:I} f_i}\downarrow & & \downarrow{\scriptstyle g \in \mathcal{R}} \\ \bigsqcup_{i:I} Y_i & \longrightarrow & B \end{array} \quad \text{gives} \quad \begin{array}{ccc} X_i & \longrightarrow & A \\ {\scriptstyle f_i \in \mathcal{L}}\downarrow & \overset{l_i}{\nearrow} & \downarrow{\scriptstyle g \in \mathcal{R}} \\ Y_i & \longrightarrow & B \end{array}$$

The lift is $\bigsqcup_{i:I} l_i$!

Closure Under Coproducts

Suppose all $f_i \in \mathcal{L}$ for $i : I$, is $\bigsqcup_{i:I} f_i \in \mathcal{L}$?

$$\begin{array}{ccc} \bigsqcup_{i:I} X_i & \longrightarrow & A \\ {\scriptstyle \bigsqcup_{i:I} f_i}\downarrow & & \downarrow{\scriptstyle g \in \mathcal{R}} \\ \bigsqcup_{i:I} Y_i & \longrightarrow & B \end{array} \quad \text{gives} \quad \begin{array}{ccc} X_i & \longrightarrow & A \\ {\scriptstyle f_i \in \mathcal{L}}\downarrow & \overset{l_i}{\nearrow} & \downarrow{\scriptstyle g \in \mathcal{R}} \\ Y_i & \longrightarrow & B \end{array}$$

The lift is $\bigsqcup_{i:I} l_i$! Yes, but

$$\prod_{i:I} \left\| \sum_{l_i : Y_i \to A} l_i \text{ fills } i\text{-th diagram} \right\|$$

$$\not\Rightarrow \left\| \sum_{L : \bigsqcup_{i:I} Y_i \to A} \prod_{i:I} \mathtt{in}_i^{\sqcup} \cdot L \text{ fills } i\text{-th diagram} \right\|$$

The Axiom of Choice

$$\prod_{i:I} \left\| \sum_{l_i : Y_i \to A} l_i \text{ fills } i\text{-th diagram} \right\|$$

$$\nRightarrow \left\| \sum_{L : \bigsqcup_{i:I} Y_i \to A} \prod_{i:I} \mathtt{in}_i^{\sqcup} \cdot L \text{ fills } i\text{-th diagram} \right\|$$

The Axiom of Choice

$$\left(\prod_{x:X} \left\| \sum_{l_x : L(x)} P(x, l_x) \right\| \right) \to \left\| \sum_{l : \prod_{x:X} L(x)} \prod_{x:X} P(x, l(x)) \right\|$$

The Axiom of Choice

$$\prod_{i:I} \left\| \sum_{l_i : Y_i \to A} l_i \text{ fills } i\text{-th diagram} \right\|$$

$$\nRightarrow \left\| \sum_{L : \bigsqcup_{i:I} Y_i \to A} \prod_{i:I} \mathtt{in}_i^{\sqcup} \cdot L \text{ fills } i\text{-th diagram} \right\|$$

The Axiom of Choice

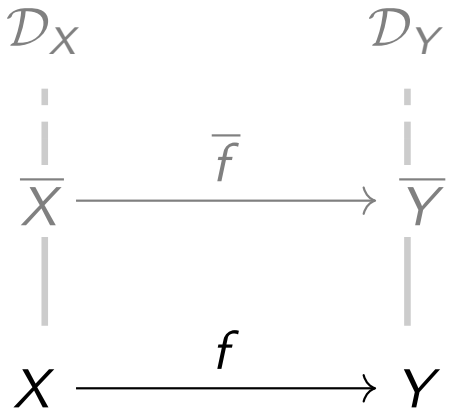
$$\left(\prod_{x:X} \left\| \sum_{l_x : L(x)} P(x, l_x) \right\| \right) \to \left\| \sum_{l : \prod_{x:X} L(x)} \prod_{x:X} P(x, l(x)) \right\|$$

Cannot prevent use of Axiom of Choice because

$$\mathcal{L} = {}^{\square}\mathcal{R}$$

Outline

Displayed Categories



Definition

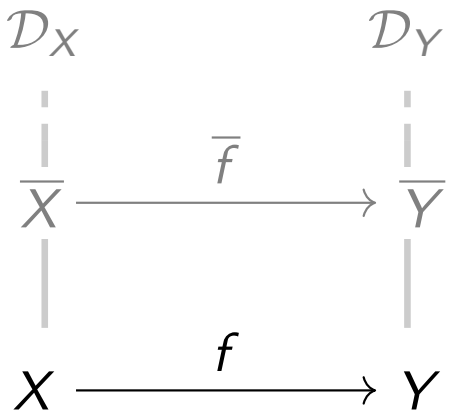
A *displayed category* $\mathcal{D}$ over a category $\mathcal{C}$ consists of the following data.

- For each $X : \mathcal{C}$ a family $\mathcal{D}_X$ of *displayed objects over* X.
- For each $f : X \to Y$ and $\overline{X} : \mathcal{D}_X$ and $\overline{Y} : \mathcal{D}_Y$ a family $\overline{X} \to_f \overline{Y}$ of *displayed morphisms over* f.
- A displayed identity morphism $1_{\overline{X}} : \overline{X} \to_{\mathrm{id}_X} \overline{X}$ and composition

$$\left(\overline{X} \to_f \overline{Y}\right) \times \left(\overline{Y} \to_g \overline{Z}\right) \to \left(\overline{X} \to_{f \cdot g} \overline{Z}\right),$$

such that the appropriate relations hold.

Displayed Categories



Definition

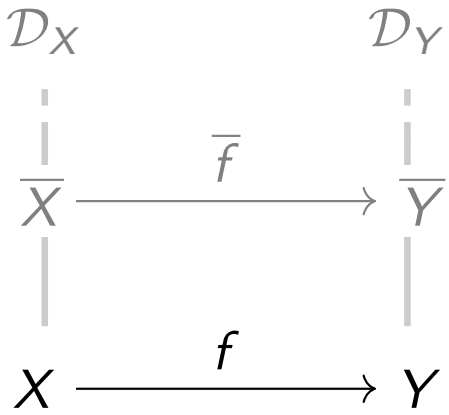
For a displayed category $\mathcal{D}$ over $\mathcal{C}$ we define the *total category* $\mathcal{D}^{\text{tot}}$ with objects

$$\sum_{X:\mathcal{C}} \mathcal{D}_X$$

and morphisms between $(X, \overline{X})$ and $(Y, \overline{Y})$

$$\sum_{f:X\to Y} \overline{X} \to_f \overline{Y}.$$

Displayed Categories



Definition

A *section from* $\mathcal{C}$ *to* $\mathcal{D}$ consists of

- A dependent function of objects
$$F : \prod_{X:\mathcal{C}} \mathcal{D}_X.$$
- A corresponding dependent function of morphisms
$$F : \prod_{f:X\to Y} F(X) \to_f F(Y).$$

such that the appropriate relations hold.

Displayed Categories

Definition

The arrow category $\mathcal{C}$ is a category $\mathcal{C}^{\mathbf{2}}$ with (∼350LoC)

objects $f : X \to Y$

morphisms

$$\begin{array}{ccc} X & \longrightarrow & A \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ Y & \longrightarrow & B \end{array}$$

Displayed Categories

Definition

The arrow category $\mathcal{C}$ is a category $\mathcal{C}^{\mathbf{2}}$ with (∼350LoC)

objects $f : X \to Y$

morphisms

$$\begin{array}{ccc} X & \longrightarrow & A \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ Y & \longrightarrow & B \end{array}$$

Example

The three category $\mathcal{C}$ is a *displayed* category $\mathcal{C}^{\mathbf{3}}$ over $\mathcal{C}^{\mathbf{2}}$ with (∼350LoC)

objects $X \xrightarrow{\lambda_f} E_f \xrightarrow{\rho_f} Y$

morphisms

$$\begin{array}{ccc} X & \longrightarrow & A \\ {\scriptstyle \lambda_f}\downarrow & & \downarrow{\scriptstyle \lambda_g} \\ E_f & \longrightarrow & E_g \\ {\scriptstyle \rho_f}\downarrow & & \downarrow{\scriptstyle \rho_g} \\ Y & \longrightarrow & B \end{array}$$

Functorial Factorizations

Definition

A functorial factorization F over a category $\mathcal{C}$ is a section of the canonical projection

$$d_1 : \mathcal{C}^{\mathbf{3}} \longrightarrow \mathcal{C}^{\mathbf{2}}.$$

```
Definition functorial_factorization (C : category) :=
    section_disp (three_disp C).
```

$$X \xrightarrow{f} Y \quad \mapsto \quad X \xrightarrow{\lambda_f} E_f \xrightarrow{\rho_f} Y$$

Objects of category $\mathbf{Ff}_{\mathcal{C}}$ (∼350LoC)

Functorial Factorizations

Example

In any category with binary coproducts, there is a functorial factorization mapping $f : X \to Y$ to

$$X \xrightarrow{\mathtt{in}^{\sqcup}_X} X \sqcup Y \xrightarrow{f \sqcup \mathsf{id}_Y} Y.$$

Functorial Factorizations

Get endofunctors $\mathcal{C}^{\mathbf{2}} \longrightarrow \mathcal{C}^{\mathbf{2}}$

$$L : f \mapsto \lambda_f \qquad \text{and} \qquad R : f \mapsto \rho_f$$

Remark

We have

$$\operatorname{dom} \lambda_f := \operatorname{dom} f \qquad \text{and} \qquad \operatorname{cod} \rho_f := \operatorname{cod} f$$

so

$$f =_{X \to Y} L(f) \cdot R(f)$$

is well-typed and trivial.

```
Lemma LR_compatibility {C : category} (F : Ff C) :
  ∏ (f : arrow C), (fact_L F f) · (fact_R F f) = f.
Proof.
  intro.
  apply three_comp.
Qed.
```

Naive Definition

$$\left\{ F : \mathcal{C}^{\mathbf{2}} \longrightarrow \mathcal{C}^{\mathbf{3}} \;\middle|\; F \cdot d_1 = \mathrm{id}_{\mathcal{C}^{\mathbf{2}}} \right\}$$

```
Definition functorial_factorization' (C : category) :=
  ∑ F : (arrow C ⟶ three C),
    F • d_1 = functor_identity (arrow C).
```

Naive Definition

$$\{ F : \mathcal{C}^{\mathbf{2}} \longrightarrow \mathcal{C}^{\mathbf{3}} \mid F \cdot d_1 = \mathrm{id}_{\mathcal{C}^{\mathbf{2}}} \}$$

```
Definition functorial_factorization' (C : category) :=
  ∑ F : (arrow C ⟶ three C),
    F • d_1 = functor_identity (arrow C).
```

Remark

We have

$$p : \mathrm{dom}\, \lambda_f =_{\mathcal{C}} \mathrm{dom}\, f \qquad \text{and} \qquad q : \mathrm{cod}\, \rho_f =_{\mathcal{C}} \mathrm{cod}\, f$$

so

$$f =_{*} L(f) \cdot R(f)$$

is ill-typed,

Naive Definition

$$\{ F : \mathcal{C}^{\mathbf{2}} \longrightarrow \mathcal{C}^{\mathbf{3}} \mid F \cdot d_1 = \mathrm{id}_{\mathcal{C}^{\mathbf{2}}} \}$$

```
Definition functorial_factorization' (C : category) :=
  ∑ F : (arrow C ⟶ three C),
    F • d_1 = functor_identity (arrow C).
```

Remark

We have

$$p : \mathrm{dom}\, \lambda_f =_{\mathcal{C}} \mathrm{dom}\, f \qquad \text{and} \qquad q : \mathrm{cod}\, \rho_f =_{\mathcal{C}} \mathrm{cod}\, f$$

so

$$f =_{*} L(f) \cdot R(f)$$

is ill-typed, and

$$f =_{X \to Y} (\mathtt{idtoiso}\ p)^{-1} \cdot L(f) \cdot R(f) \cdot \mathtt{idtoiso}\ q$$

is not trivial.

Natural Weak Factorization Systems

$$\Phi : L \Longrightarrow \mathrm{id}_{\mathcal{C}^2} \qquad \text{and} \qquad \Lambda : \mathrm{id}_{\mathcal{C}^2} \Longrightarrow R$$

$$\Phi_f := \begin{array}{ccc} X & = & X \\ {\scriptstyle \lambda_f}\downarrow & & \downarrow{\scriptstyle f} \\ E_f & \xrightarrow[\rho_f]{} & Y \end{array} \qquad \text{and} \qquad \Lambda_f := \begin{array}{ccc} X & \xrightarrow{\lambda_f} & E_f \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle \rho_f} \\ Y & = & Y \end{array}$$

$$(L, \Phi) \qquad \text{and} \qquad (R, \Lambda)$$

Natural Weak Factorization Systems

Definition

A *Natural Weak Factorization System* (NWFS) on a category $\mathcal{C}$ is given by a functorial factorizaton $F : \mathcal{C}^{\mathbf{2}} \longrightarrow \mathcal{C}^{\mathbf{3}}$, together with an extension of (R, Λ) to a monad $\mathsf{R} := (R, \Lambda, \Pi)$ and an extension of (L, Φ) to a comonad $\mathsf{L} := (L, \Phi, \Sigma)$.

Objects of displayed category $\mathbf{NWFS}_{\mathcal{C}}$ over $\mathbf{Ff}_{\mathcal{C}}$. (∼450LoC)

Natural Weak Factorization Systems

Definition

A *Natural Weak Factorization System* (NWFS) on a category $\mathcal{C}$ is given by a functorial factorizaton $F : \mathcal{C}^{\mathbf{2}} \longrightarrow \mathcal{C}^{\mathbf{3}}$, together with an extension of (R, Λ) to a monad $\mathsf{R} := (R, \Lambda, \Pi)$ and an extension of (L, Φ) to a comonad $\mathsf{L} := (L, \Phi, \Sigma)$.

Objects of displayed category $\mathbf{NWFS}_{\mathcal{C}}$ over $\mathbf{Ff}_{\mathcal{C}}$. (∼450LoC)

Definition

We define the class L-**Map** to be the class of coalgebras of L. Similarly, the class R-**Map** is the class of algebras of R.

Structure, not property

Natural Weak Factorization Systems

Example

In any category with binary coproducts, the factorization

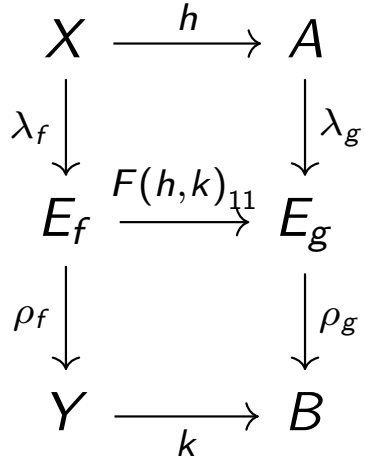
$$X \xrightarrow{\mathtt{in}^{\sqcup}_{X}} X \sqcup Y \xrightarrow{f \sqcup \mathsf{id}_Y} Y$$

admits an NWFS structure, where the R-**Map**s are the split epimorphisms and the L-**Map**s are the complemented monomorphisms.

(∼600LoC)

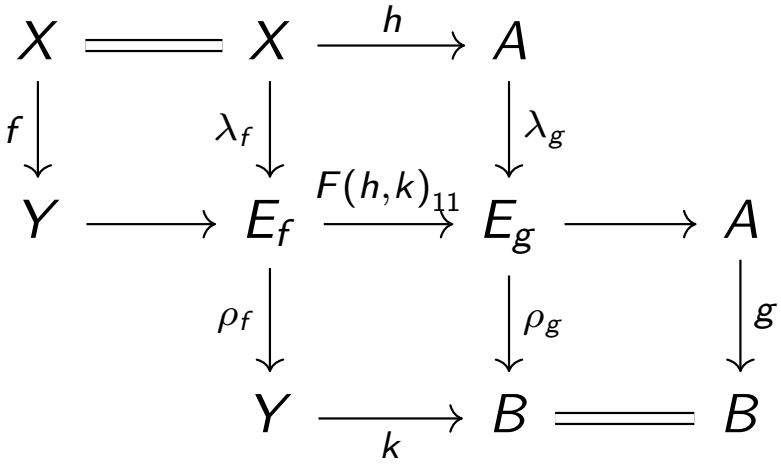
Natural Weak Factorization Systems

For an (f, g)-lifting problem (h, k) between an L-**Map** f and a R-**Map** g



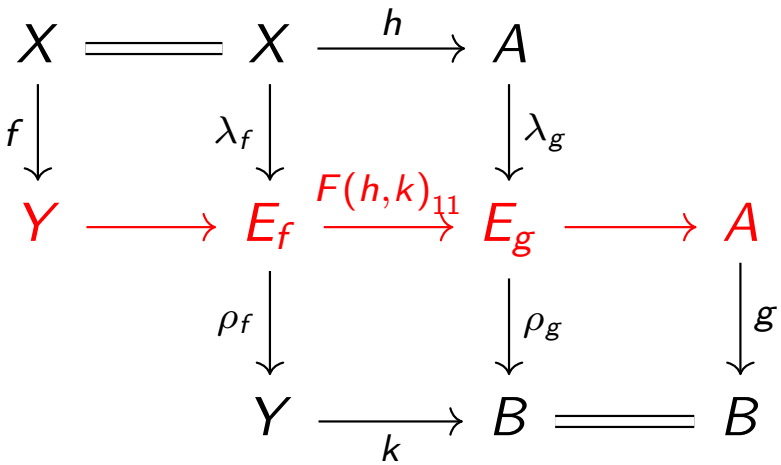
Natural Weak Factorization Systems

For an (f, g)-lifting problem (h, k) between an L-**Map** f and a R-**Map** g



Natural Weak Factorization Systems

For an (f, g)-lifting problem (h, k) between an L-**Map** f and a R-**Map** g



Outline

Idea

- Start with 'well-behaved' morphism class J.
- Get WFS with $J \subseteq \mathcal{L}$ and $\mathcal{R} = J^{\square}$.
- Inductive, left map is always in $\mathcal{L}$, right map has to be 'fixed'.
- Fix by iteratively glueing cells to domain of right map.
- 'Well-behavedness' allows us to 'cut off' the transfinite construction.

Example

In **TOP** or **SSET**,

$$J = \{ S^n \hookrightarrow D_{n+1} \}.$$

Idea

- Start with 'well-behaved' morphism class J.
- Get WFS with $J \subseteq \mathcal{L}$ and $\mathcal{R} = J^{\square}$.
- Inductive, left map is always in $\mathcal{L}$, right map has to be 'fixed'.
- Fix by iteratively glueing cells to domain of right map.
- 'Well-behavedness' allows us to 'cut off' the transfinite construction.

Example

In **TOP** or **SSET**,

$$J = \{ S^n \hookrightarrow D_{n+1} \}.$$

Idea

- Start with 'well-behaved' morphism class J.
- Get WFS with $J \subseteq \mathcal{L}$ and $\mathcal{R} = J^{\square}$.
- Inductive, left map is always in $\mathcal{L}$, right map has to be 'fixed'.
- Fix by iteratively glueing cells to domain of right map.
- 'Well-behavedness' allows us to 'cut off' the transfinite construction.

Example

In **TOP** or **SSET**,

$$J = \{ S^n \hookrightarrow D_{n+1} \}.$$

Idea

- Start with 'well-behaved' morphism class J.
- Get WFS with $J \subseteq \mathcal{L}$ and $\mathcal{R} = J^{\square}$.
- Inductive, left map is always in $\mathcal{L}$, right map has to be 'fixed'.
- Fix by iteratively glueing cells to domain of right map.
- 'Well-behavedness' allows us to 'cut off' the transfinite construction.

Example

In **TOP** or **SSET**,

$$J = \{ S^n \hookrightarrow D_{n+1} \}.$$

Idea

- Start with 'well-behaved' morphism class J.
- Get WFS with $J \subseteq \mathcal{L}$ and $\mathcal{R} = J^{\square}$.
- Inductive, left map is always in $\mathcal{L}$, right map has to be 'fixed'.
- Fix by iteratively glueing cells to domain of right map.
- 'Well-behavedness' allows us to 'cut off' the transfinite construction.

Example

In **TOP** or **SSET**,

$$J = \{ S^n \hookrightarrow D_{n+1} \}.$$

Idea

- Start with 'well-behaved' morphism class J.
- Get WFS with $J \subseteq \mathcal{L}$ and $\mathcal{R} = J^{\square}$.
- Inductive, left map is always in $\mathcal{L}$, right map has to be 'fixed'.
- Fix by iteratively glueing cells to domain of right map.
- 'Well-behavedness' allows us to 'cut off' the transfinite construction.

Example

In **TOP** or **SSET**,

$$J = \{ S^n \hookrightarrow D_{n+1} \}.$$

The Construction

The first step

Let $f : X \to Y$. We inductively define factorization

$$X \xrightarrow{\lambda_\alpha^f} Z_\alpha^f \xrightarrow{\rho_\alpha^f} Y$$

First step:

$$X \xrightarrow{\lambda_0^f} X \xrightarrow{f} Y$$

The Construction

The inductive step

At step α, set S to be the set of lifting problems

$$\begin{array}{ccc} A & \longrightarrow & Z^f_\alpha \\ {\scriptstyle g \in J}\downarrow & & \downarrow{\scriptstyle \rho^f_\alpha} \\ B & \longrightarrow & Y \end{array}$$

Next step is pushout

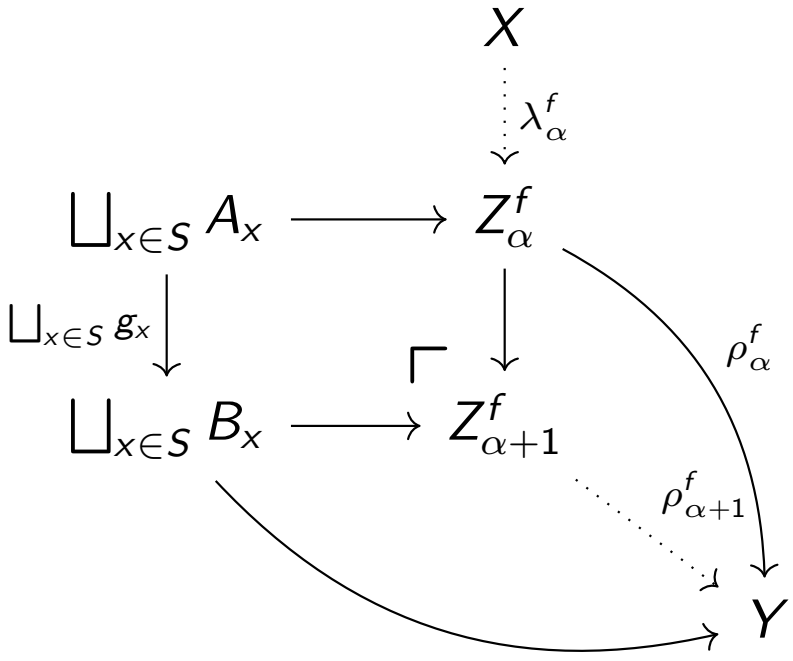
The Construction

The inductive step

At step α, set S to be the set of lifting problems

$$\begin{array}{ccc} A & \longrightarrow & Z^f_\alpha \\ {\scriptstyle g \in J}\downarrow & & \downarrow{\scriptstyle \rho^f_\alpha} \\ B & \longrightarrow & Y \end{array}$$

Next step is pushout



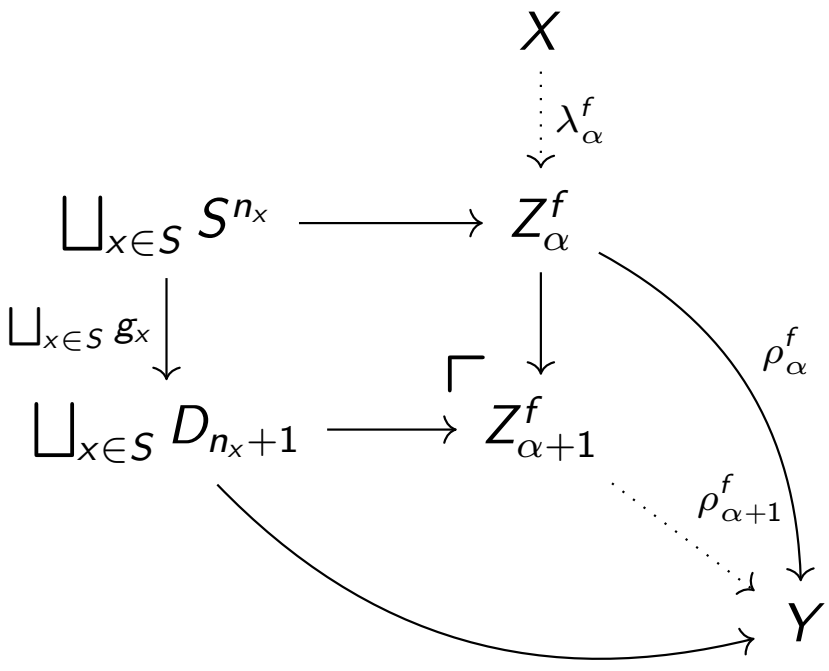
The Construction

The inductive step

At step α, set S to be the set of lifting problems

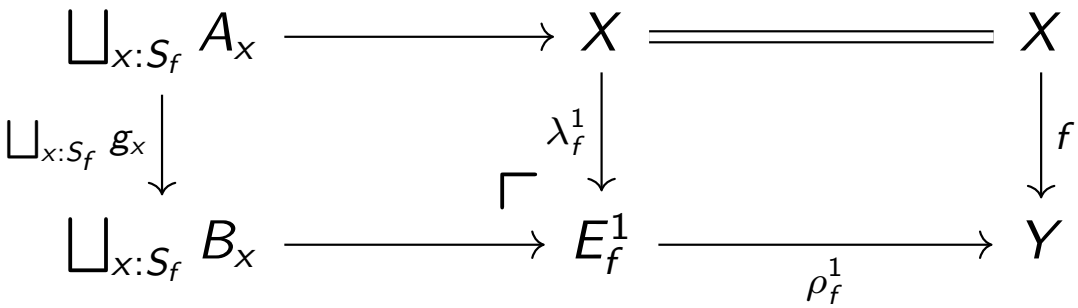
$$\begin{array}{ccc} S^n & \longrightarrow & Z^f_\alpha \\ {\scriptstyle g \in J}\downarrow & & \downarrow{\scriptstyle \rho^f_\alpha} \\ D_{n+1} & \longrightarrow & Y \end{array}$$

Next step is pushout



Algebraic Small Object Argument

First step

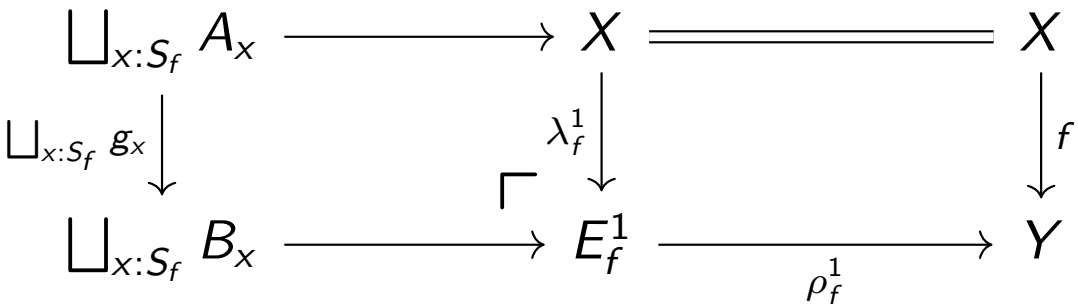


Get functor (∼500LoC)

$$L^1 : f \mapsto \lambda_f^1$$

Algebraic Small Object Argument

First step

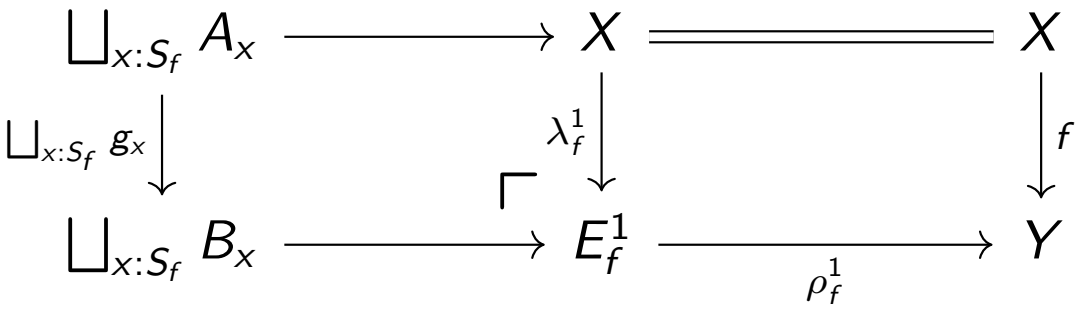


Get comonad (∼1000LoC)

$$L^1 : f \mapsto \lambda_f^1$$

Algebraic Small Object Argument

First step

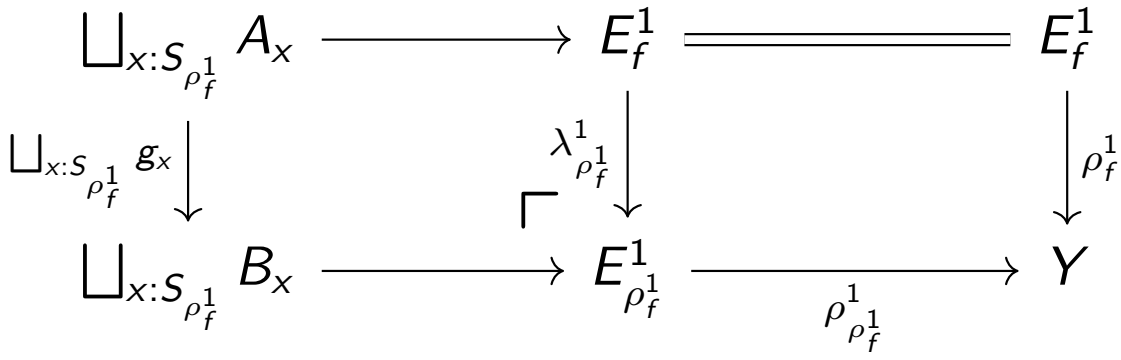


Get comonad (∼1000LoC)

$$L^1 : f \mapsto \lambda_f^1$$

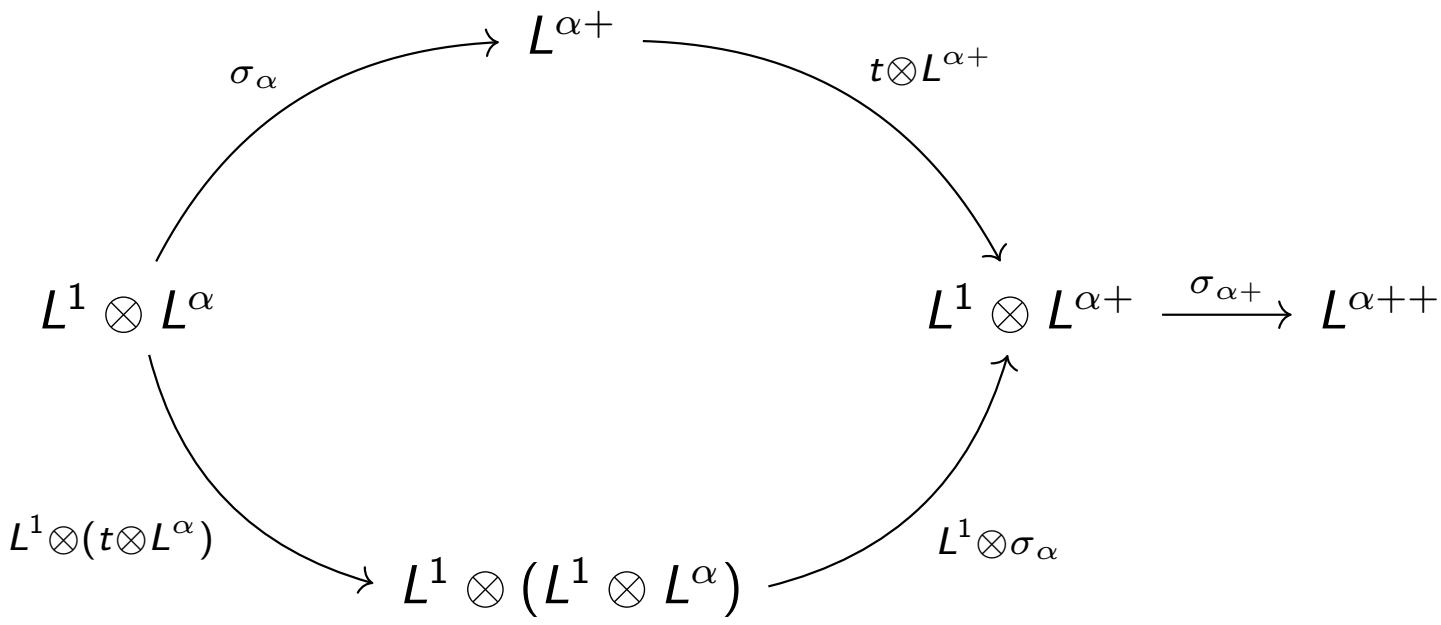
Monoidal structure to get (∼2000LoC + frustration)

$$L^1 \otimes L^1 : f \mapsto \lambda_f^1 \cdot \lambda^1_{\rho_f^1}$$



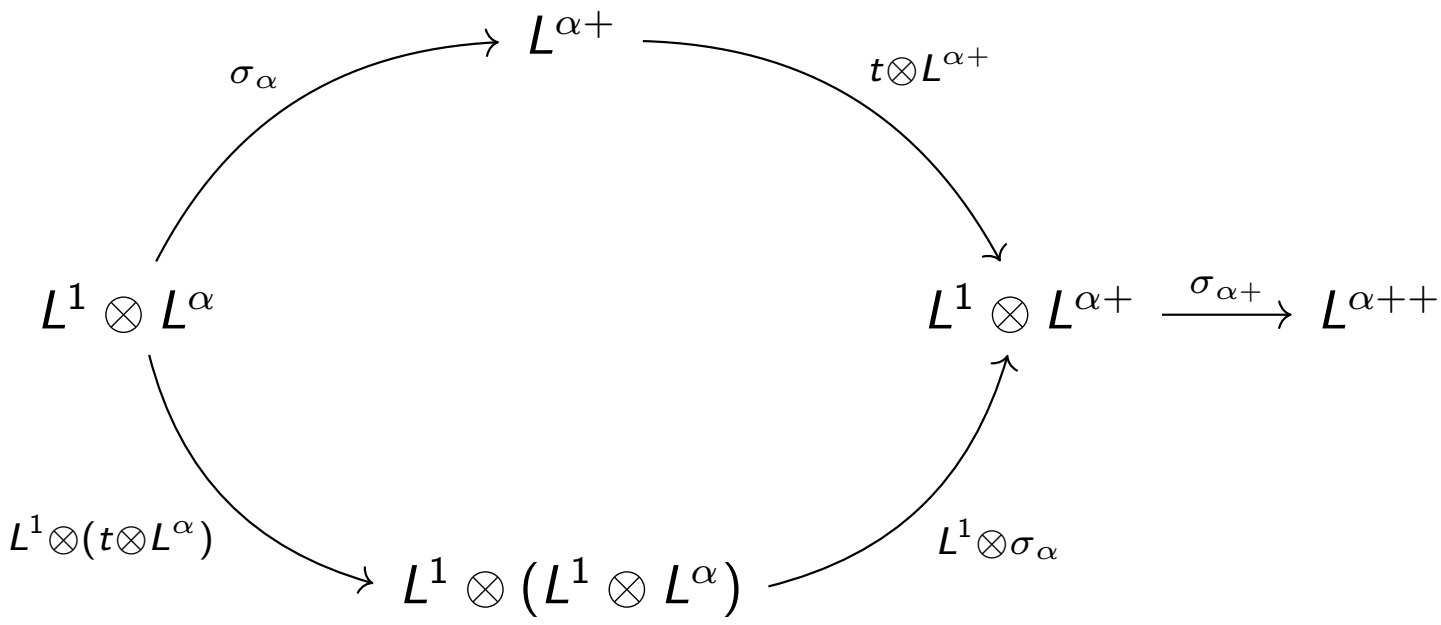
Fixing Duplicate Cells

Successor step



Fixing Duplicate Cells

Successor step



- No duplicate cells.
- Algebraically coherent steps.
- Simple convergence condition.
- Generalized from transfinite construction by Kelly.
- Rest of construction: >6000LoC, details, intuition, modifications.

Outline

Conclusion

Conclusion:

- Formalization of ∼15000 LoC in 30 files.
- Found constructive issues in classical theory.
- Formalized existence of NWFS.
- Argument works for (HoTT) categories.
- No use of *Univalence Axiom*.
- Argument works for modern, weaker notion of monoidal categories.
- Made part of argument more direct and constructive.

Future work:

- Formalize other smallness requirement.
- Formalize *cofibrantly generatedness*.
- Formalize general theory of ordinals.
- Lean formalization.
- Model 2-categories.

Conclusion

Conclusion:

- Formalization of ∼15000 LoC in 30 files.
- Found constructive issues in classical theory.
- Formalized existence of NWFS.
- Argument works for (HoTT) categories.
- No use of *Univalence Axiom*.
- Argument works for modern, weaker notion of monoidal categories.
- Made part of argument more direct and constructive.

Future work:

- Formalize other smallness requirement.
- Formalize *cofibrantly generatedness*.
- Formalize general theory of ordinals.
- Lean formalization.
- Model 2-categories.

Thank you for your attention!

Proof finished.

Smallness

Definition

Suppose $\mathcal{C}$ is a category with all small colimits, and J is a class of morphisms of $\mathcal{C}$. Let A be an object of $\mathcal{C}$ and κ a cardinal. We say that A is *κ-small relative to J* if, for all κ-filtered ordinals γ and γ-sequences

$$X_0 \to X_1 \to \ldots \to X_\alpha \to \ldots$$

where each map $X_\alpha \to X_{\alpha+1}$ is in J for $\alpha + 1 < \gamma$, the map of sets

$$\mathrm{colim}_{\alpha<\gamma}(A \to X_\alpha) \to (A \to \mathrm{colim}_{\alpha<\gamma} X_\alpha)$$

is an isomorphism. We say that A is *small relative to J* if it is κ-small relative to J for some cardinal κ. We say that A is *small* if it is small relative to all morphisms in $\mathcal{C}$.

'A map into a long enough composition of maps in J factors through some stage of this composition'

Smallness

Definition

Let $\mathcal{C}$ be a category and X and object in $\mathcal{C}$. Then X is called *presentable* if and only if the covariant homset functor

$$\mathrm{Hom}(X, -) : \mathcal{C} \longrightarrow \mathbf{SET}$$

is ω-small.

Formalized: Every $g \in J$ is presentable in $\mathcal{C}^{\mathbf{2}}$.
Literature:

- Every object $\mathrm{dom}\, g$ for $g \in J$ is presentable in $\mathcal{C}$
- *Or:* For every $\mathrm{dom}\, g$ there is an α for which $\mathrm{Hom}(\mathrm{dom}\, g, -)$ preserves α-filtered unions of $\mathcal{M}$-subobjects for some proper cowellpowered strong factorization system $(\mathcal{E}, \mathcal{M})$.

Second requirement: 'presentable relative to subspace inclusions' for **TOP**.

Rest of the Construction

Presentability of J implies that L^1 is ω-small, preserving connected colimits like those of chains or coequalizers. This shows that

$$L^\infty \to L^1 \otimes L^\infty \to L^{\infty+1}$$

is an isomorphism. We get

$$L^1 \otimes L^\infty \to L^\infty$$

allowing us to inductively construct a monoid (or get a monoid from an adjunction like Garner does)

$$L^\infty \otimes L^\infty \to L^\infty,$$

providing the extension of R^∞ to a monad.

Examples

Example

Every category $\mathcal{C}$ that is both complete and cocomplete admits the *trivial model structure*, where the weak equivalences are the class `isos` of isomorphisms and both the fibrations and the cofibrations are the class `univ` of all maps.

Example

The example on **SET**, with $J = \{ \emptyset \to * \}$, generates the NWFS with underlying factorization

$$X \xrightarrow{f} Y \mapsto X \xrightarrow{\mathtt{in}_X^{\sqcup}} X \sqcup Y \xrightarrow{f \sqcup \mathrm{id}_Y} Y.$$

Examples

Example

Perhaps the most well-known example of a model structure is on the category **TOP** of topological spaces, where the fibrations are the Serre fibrations, the cofibrations are retracts of relative CW-complexes and weak equivalences are the *weak* homotopy equivalences. This model structure is referred to as the *classical model structure on* **TOP**. Generated by

$$J := \{ j_n : S^n \hookrightarrow D_{n+1} \mid n = -1, 0, 1, \dots \}$$

Example

The other examples of WFSs on **TOP** also form a model structure: with homotopy equivalences as weak equivalences, Hurewicz fibrations as fibrations and the *closed Hurewicz cofibrations* as cofibrations. This model structure is sometimes referred to as the *Hurewicz* or *Strøm model structure*.

Examples

Example

In the category of **SSET**, there is a model structure with weak homotopy equivalences as weak equivalences, the class of *Kan fibrations* as fibrations, and with cofibrations the class of monomorphisms $f : X \to Y$ which are levelwise injections. This model structure is known as the *Quillen model structure* on **SSET**, and can in fact be used to present type theory itsel. Generated by

$$J := \{ j_n : S^n \hookrightarrow D_{n+1} \mid n = -1, 0, 1, \dots \}$$

There are many more examples of model structures, such as model structures on R-Mod, the category of R-Modules for a Frobenius ring R, or model structures on the category of R chain complexes, where R is any ring.