Homotopy invariance

Question

Suppose two model categories $\mathcal{M}$, $\mathcal{N}$ present the same $(\infty, 1)$-category $\mathcal{C}$. Do they have the same internal type theory?

1. All type-theoretic operations are homotopy invariant (represent well-defined $(\infty, 1)$-categorical operations).
2. Therefore, any type-theoretic construction performed on equivalent data in $\mathcal{M}$ and $\mathcal{N}$ yields equivalent results.
3. All type-theoretic data is terms in (dependent) types, i.e. sections of fibrations. If all objects in $\mathcal{M}$ and $\mathcal{N}$ are cofibrant, any “section” in $\mathcal{C}$ can be represented in both $\mathcal{M}$ and $\mathcal{N}$.
4. The only trouble is with asserting computational equalities, e.g. “let $G$ be a group with computationally associative multiplication”. If we stick with properties that can be expressed in the type theory, we are fine.
Recall: positive types are characterized by their introduction rules.

In fact, any choice of introduction rule(s) determines a positive type in an algorithmic way.

- The derived eliminator literally does a case analysis on the introduction rules.
- We call these introduction rules constructors.

**Example (Coproduct types)**

- **Introduction**: \(\text{inl}: A \to A + B\) and \(\text{inr}: B \to A + B\)
- **Elimination**: If \((x: A) \vdash (c_A : C(\text{inl}(x)))\) and \((y: B) \vdash (c_B : C(\text{inr}(y)))\), then for \(p: A + B\) we have \(\text{case}(p, c_A, c_B) : C(p)\).

**Example (Empty type)**

- **Introduction**: 
- **Elimination**: If (nothing), then for \(p: \emptyset\) we have \(\text{abort}(p) : C(p)\).
The natural numbers are a positive type.

1. **Formation**: There is a type $\mathbb{N}$.
2. **Introduction**: $0 : \mathbb{N}$, and $(x : \mathbb{N}) \vdash (s(x) : \mathbb{N})$.

A new feature: the *input* of the constructor “$s$” involves something of the type $\mathbb{N}$ being defined!

We intend, of course, that all elements of $\mathbb{N}$ are generated by *successively* applying constructors.

$$0, s(0), s(s(0)), s(s(s(0))), \ldots$$

The natural numbers

1. **Formation**: There is a type $\mathbb{N}$.
2. **Introduction**: $0 : \mathbb{N}$, and $(x : \mathbb{N}) \vdash (s(x) : \mathbb{N})$.
3. **Elimination?** If $c_0 : C(0)$ and $(x : \mathbb{N}) \vdash (c_s : C(s(x)))$, then for $p : \mathbb{N}$ we have $\text{match}(p, c_0, c_s) : C(p)$.

But this is not much good; we need to **recurse**.

3. **Elimination**: If $c_0 : C(0)$ and

$$\vdash (x : \mathbb{N}), (r : C(x)) \vdash (c_s : C(s(x)))$$

then for $p : \mathbb{N}$ we have $\text{rec}(p, c_0, c_s) : C(p)$.

The variable $r$ represents the result of the recursive call at $x$, to be used the computation $c_s$ of the value at $s(x)$.
Example: Addition

We define addition by recursion on the first input.

\[
\begin{align*}
\text{plus}(0, m) & := m \\
\text{plus}(s(n), m) & := s(\text{plus}(n, m))
\end{align*}
\]

In terms of the rec eliminator, this is

\[
(n : \mathbb{N}), (m : \mathbb{N}) \vdash \text{plus}(n, m) := \text{rec}(n, m, s(r))
\]

- When \( n = 0 \), the result is \( m \).
- When \( n \) is a successor \( s(x) \), the result is \( s(r) \).
  (As before, \( r \) is the result of the recursive call at \( x \).)

The natural numbers

1. **Formation:** There is a type \( \mathbb{N} \).
2. **Introduction:** \( 0 : \mathbb{N} \), and \((x : \mathbb{N}) \vdash (s(x) : \mathbb{N})\).
3. **Elimination:** If \( c_0 : C(0) \) and

\[
(x : \mathbb{N}), (r : C(x)) \vdash (c_s : C(s(x)))
\]

then for \( p : \mathbb{N} \) we have \( \text{rec}(p, c_0, c_s) : C(p) \).
4. **Computation:**
   - \( \text{rec}(0, c_0, c_s) \) computes to \( c_0 \).
   - \( \text{rec}(s(n), c_0, c_s) \) computes to \( c_s \) with \( n \) substituted for \( x \) and
     \( \text{rec}(n, c_0, c_s) \) substituted for \( r \).
Computing an addition

\[
\text{plus}(ss, sss) := \text{rec}(ss, sss, s(r))
\]
\[
\leadsto s\left(\text{rec}(0, sss, s(r))\right)
\]
\[
\leadsto s(s(\text{rec}(0, sss, s(r))))
\]
\[
\leadsto s(s(sss)) = sssss
\]

Other recursive inductive types

Generalized positive types of this sort are called inductive types.

**Example (Lists)**

For any type \(A\), there is a type \(\text{List}(A)\), with constructors

\[
\vdash \text{nil}: \text{List}(A)
\]
\[
(a: A), (\ell: \text{List}(A)) \vdash (\text{cons}(a, \ell): \text{List}(A))
\]

Functional programming is built on defining functions by recursion over inductive datatypes.

\[
\text{length}(\text{nil}) := 0
\]
\[
\text{length}(\text{cons}(a, \ell)) := s(\text{length}(\ell))
\]

This is defined using the eliminator for \(\text{List}(A)\).
Proof by induction

3 If \( c_0 : C(0) \) and

\[
(x : \mathbb{N}), (r : C(x)) \vdash (c_s : C(s(x)))
\]

then for \( p : \mathbb{N} \) we have \( \text{rec}(p, c_0, c_s) : C(p) \).

When \( C \) is a predicate, this is just proof by induction.

<table>
<thead>
<tr>
<th>types</th>
<th>propositions</th>
</tr>
</thead>
<tbody>
<tr>
<td>programming</td>
<td>proving</td>
</tr>
<tr>
<td>recursion</td>
<td>induction</td>
</tr>
</tbody>
</table>

Conclusion

Proof by induction is not something special about the natural numbers; it applies to any inductive type.

Recursively defined types

We can define dependent types as Type-valued recursive functions.

Theorem

\( 0 \neq 1 \).

Proof.

Define \( P : \mathbb{N} \to \text{Type} \) by “recursion”:

\[
P(0) := 1 \\
P(s(n)) := \emptyset
\]

- Suppose \( p : (0 = 1) \).
- Since \( * : P(0) \), we have \( \text{trans}(p, *) : P(1) \equiv \emptyset \).
- Thus, \( \lambda p. \text{trans}(p, tt) : ((0 = 1) \to \emptyset) \equiv \neg(0 = 1) \).
Example: Truncation

**Definition**

An $\infty$-groupoid is $n$-truncated if it has no nontrivial $k$-morphisms for any $k > n$.

- $h$-sets are 0-truncated.
- $A$ is $(n+1)$-truncated $\iff$ each $(x = y)$ is $n$-truncated.
- $A$ is an $h$-set $\iff$ each $(x = y)$ is an $h$-prop.
  Thus, it makes sense to call $h$-props “$(−1)$-truncated”.
- $A$ is an $h$-prop $\iff$ each $(x = y)$ is contractible.
  Thus, we call contractible spaces “$(−2)$-truncated”.
- After this, it’s “turtles all the way down”: $(−3)$-truncated is the same as $(−2)$-truncated.
- (Voevodsky) $h$-level $n$ means $(n−2)$-truncated.

\[ \text{isHlevel}(0, A) := \text{isContr}(A) \]
\[ \text{isHlevel}(s(n), A) := \prod_{x : A} \prod_{y : A} \text{isHlevel}(n, (x = y)) \]

**Inductive families**

We can define dependent types inductively as well.

**Example (Vectors)**

For any $A$ there is a dependent type $\text{Vec}(A) : \mathbb{N} \to \text{Type}$, with constructors

\[ \vdash \text{nil} : \text{Vec}(A, 0) \]
\[ (a : A), (n : \mathbb{N}), (\ell : \text{Vec}(A, n)) \vdash (\text{cons}(a, \ell) : \text{Vec}(A, s(n))) \]

(We build the length of a list into its type.)

**Example (Equality!)**

For any $A$ there is a dependent type $\text{Eq}_A : A \times A \to \text{Type}$, with constructor

\[ (a : A) \vdash (\text{refl}_a : \text{Eq}_A(a, a)) \]
The positive type \( \mathbb{N} \) should have a left universal property.

**Definition**

A natural numbers object is \( \mathbb{N} \) with \( 0 : 1 \to \mathbb{N} \), \( s : \mathbb{N} \to \mathbb{N} \), s.t.

- For any object \( X \) with \( 0_X : 1 \to X \) and \( s_X : X \to X \), there is a unique \( r : \mathbb{N} \to X \) such that

\[
\begin{array}{ccc}
1 & \to & \mathbb{N} \\
0_X & \downarrow & \downarrow r \\
X & \to & X \\
\end{array}
\]

\( s \mathbb{N} \)

\( \Rightarrow \) an initial object in the category of triples \((X, 1 \to X, X \to X)\).

natural numbers type \( \mathbb{N} \) \( \iff \) natural numbers object

**Algebras for endofunctors**

Let \( F \) be a functor from a category to itself.

**Definition**

An \( F \)-algebra is an object \( X \) with a morphism \( x : F(X) \to X \). An \( F \)-algebra map is a map \( f : X \to Y \) such that

\[
\begin{array}{ccc}
F(X) & \xrightarrow{F(f)} & F(Y) \\
\downarrow x & & \downarrow y \\
X & \xrightarrow{f} & Y \\
\end{array}
\]

An initial \( F \)-algebra is an initial object in the category of \( F \)-algebras and \( F \)-algebra maps.
Inductive types and endofunctors

\[
\text{inductive types} \quad \leftrightarrow \quad \text{initial algebras for endofunctors}
\]

<table>
<thead>
<tr>
<th>Inductive type</th>
<th>Endofunctor</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbb{N} )</td>
<td>( F(X) := 1 + X )</td>
</tr>
<tr>
<td>( \text{List}(A) )</td>
<td>( F(X) := 1 + (A \times X) )</td>
</tr>
<tr>
<td>( A + B )</td>
<td>( F(X) := A + B )</td>
</tr>
</tbody>
</table>

\( (a \text{ constant endofunctor}) \)

The eliminator directly asserts only weak initiality, but using the dependent eliminator one can prove:

**Theorem (Awodey–Gambino–Sojakova)**

Any inductive type \( W \) is a homotopy initial \( F \)-algebra: the space of \( F \)-algebra maps \( W \to X \) is contractible.

**Constructing initial algebras**

We also have:

**Theorem**

If \( F \) is an accessible endofunctor of a locally presentable category, then there exists an initial \( F \)-algebra.

**Sketch of proof.**

Take the colimit of the transfinite sequence

\[
\emptyset \to F(\emptyset) \to F(F(\emptyset)) \to \cdots
\]
Higher inductive types

Idea

- **Inductive types** are a good way to build sets: we specify the elements of a set by giving constructors.
- To build a space (or ∞-groupoid), we need to specify not only elements, but paths and higher paths.
- The iterative construction of initial algebras looks a lot like the small object argument.
- Is there an analogous notion of higher inductive type that described more general cell complexes?

Example

The circle $S^1$ should be inductively defined by two constructors

\[
\text{base} : S^1 \quad \text{and} \quad \text{loop} : (\text{base} = \text{base})
\]

Can we make sense of this?

The circle (first try)

1. **Formation:** There is a type $S^1$.
2. **Introduction:** $\text{base} : S^1$ and $\text{loop} : (\text{base} = \text{base})$.
3. **Elimination:** Given $b : C$ and $\ell : (b = b)$, for any $p : S^1$ we have $\text{match}(p, b, \ell) : C$.
4. **Computation:** $\text{match}($base, $b$, $\ell$) computes to $b$, and $\text{map}(\text{match}(\text{--} , b, \ell), \text{loop})$ computes to $\ell$.

What about a dependent eliminator?
Dependent loops

As hypotheses of the dependent eliminator for $S^1$, we need

1. A point $b : C\text{(base)}$.
2. A path $\ell$ from $b$ to $b$ lying over “loop”.

Dependent loops

As hypotheses of the dependent eliminator for $S^1$, we need

1. A point $b : C\text{(base)}$.
2. A path $\ell'$ from $b$ to $b$ lying over “loop”.
Dependent loops

As hypotheses of the dependent eliminator for $S^1$, we need

1. A point $b : C(\text{base})$.
2. A path $\ell$ from $b$ to $b$ lying over “loop”.

Dependent loops

As hypotheses of the dependent eliminator for $S^1$, we need

1. A point $b : C(\text{base})$.
2. A path $\ell$ from $b$ to $b$ lying over “loop”.
As hypotheses of the dependent eliminator for $S^1$, we need

1. A point $b : C(\text{base})$.
2. A path $\ell$ from $b$ to $b$ lying over “loop”.

The circle (final version)

1. **Formation:** There is a type $S^1$.
2. **Introduction:** base : $S^1$ and loop : (base = base).
3. **Elimination:** Given $b : C(\text{base})$ and $\ell : (\text{transport}(\text{loop}, b) = b)$, for any $p : S^1$ we have match($p, b, \ell$) : $C(p)$.
4. **Computation:** match(base, $b, \ell$) computes to $b$, and map(match($-$, $b, \ell$), loop) computes to $\ell$. 
The Interval

Example

The interval $I$ is an inductive type with three constructors:

\[
\begin{align*}
\text{zero} &: I \\
\text{one} &: I \\
\text{segment} &: (\text{zero} = \text{one})
\end{align*}
\]

- Unsurprisingly, this type is provably contractible.
- But surprisingly, it is not useless; it implies function extensionality.

The 2-sphere

Example

The 2-sphere $S^2$ has two constructors:

\[
\begin{align*}
\text{base2} &: S^2 \\
\text{loop2} &: (\text{refl}_{\text{base2}} = \text{refl}_{\text{base2}})
\end{align*}
\]

OR:

\[
\begin{align*}
\text{northpole} &: S^2 \\
\text{southpole} &: S^2 \\
\text{greenwich} &: (\text{northpole} = \text{southpole}) \\
\text{dateline} &: (\text{northpole} = \text{southpole}) \\
\text{east} &: (\text{greenwich} = \text{dateline}) \\
\text{west} &: (\text{greenwich} = \text{dateline})
\end{align*}
\]

etc...
The torus

Example

The torus $T^2$ has four constructors:

- $pt : T^2$
- $p : (pt = pt)$
- $q : (pt = pt)$
- $surf : (p * q = q * p)$

Cylinders

Example

The cylinder $Cyl(A)$ on $A$ has three constructors:

- $(a : A) \vdash (top(a) : Cyl(A))$
- $(a : A) \vdash (bot(a) : Cyl(A))$
- $(a : A) \vdash (seg(a) : (top(a) = bot(a)))$
Homotopy pushouts

Example

The homotopy pushout of $f : A \to B$ and $g : A \to C$ has three constructors:

\[
\begin{align*}
(b : B) & \vdash \left(\text{left}(b) : \text{pushout}(f, g)\right) \\
(c : C) & \vdash \left(\text{right}(c) : \text{pushout}(f, g)\right) \\
(a : A) & \vdash \left(\text{glue}(a) : (\text{left}(f(a)) = \text{right}(g(a)))\right)
\end{align*}
\]

Suspension

Example

The suspension $\Sigma A$ of $A$ has three constructors:

\[
\begin{align*}
\text{north} : \Sigma A \\
\text{south} : \Sigma A \\
(a : A) & \vdash \left(\text{mer}(a) : (\text{north} = \text{south})\right)
\end{align*}
\]
Example

The $n$-sphere $S^n$ is defined by recursion on $n$:

\[
S^0 := 1 + 1 \\
S^{s(n)} := \Sigma(S^n)
\]

Nontriviality

Theorem

The type $S^1$ is contractible $\iff$ all types are h-sets.

Proof.

Easy; $S^1$ is the “universal loop”.

HITs by themselves don’t guarantee the homotopy theory is nontrivial. We need something else, like univalence.
\( \pi_1(S^1) \cong \mathbb{Z} \), classically

How do we prove this classically?

1. Consider the winding map \( \mathbb{R} \to S^1 \).
2. This is the universal cover of \( S^1 \).
3. Thus, its fiber over a point, namely \( \mathbb{Z} \), is \( \pi_1(S^1) \).

The universal cover of \( S^1 \)
A more homotopy-theoretic way to phrase the classical proof:

1. We have a fibration \( \mathbb{R} \to S^1 \) with fiber \( \mathbb{Z} \).
2. We have a map \( * \to S^1 \), whose homotopy fiber is \( \Omega S^1 \).
3. \( \mathbb{R} \) is contractible, so we have an equivalence \( * \simeq \mathbb{R} \) over \( S^1 \).
   By the short five lemma, the induced map on homotopy fibers is an equivalence.

\[
\begin{array}{ccc}
\Omega S^1 & \to & * \\
\sim & & \sim \\
\downarrow & & \downarrow \\
\mathbb{Z} & \to & \mathbb{R} \\
\end{array}
\]

4. In particular, \( \pi_1(S^1) \cong \mathbb{Z} \).

How can we build the fibration \( \mathbb{R} \to S^1 \) in type theory?

- A fibration over \( S^1 \) is a dependent type \( R: S^1 \to \text{Type} \).
- By the eliminator for \( S^1 \), a function \( R: S^1 \to \text{Type} \) is determined by
  - A point \( B: \text{Type} \) and
  - A path \( \ell: (B = B) \).
- By univalence, \( \ell \) is an equivalence \( B \simeq B \).
Thus we can take \( B = \mathbb{Z} \) and \( \ell \) to be “+1”.

- All that’s left to do is prove that \( \sum_x: S^1 R(x) \) is contractible.
  We can do this by “induction” on \( S^1 \).
- What we get is \( \Omega S^1 \cong \mathbb{Z} \), which is classically stronger than
  \( \pi_1(S^1) \cong \mathbb{Z} \). Here, we don’t yet have a definition of \( \pi_1 \).
Recall: $A$ is $(-1)$-truncated, or an h-prop, if

$$\prod_{x,y : A} (x = y).$$

The support of $A$, denoted $\text{supp}(A)$, is supposed to be:

- an h-prop that contains a point precisely when $A$ does.
- a reflection of $A$ into h-props.

**Support as an HIT**

**Definition (Lumsdaine)**

The support of $A$ is inductively defined by two constructors:

$$(a : A) \vdash (\text{inhab}(a) : \text{supp}(A))$$
$$((x : \text{supp}(A)), (y : \text{supp}(A)) \vdash (\text{inpath}(x, y) : (x = y)))$$

The type of inpath is precisely $\text{isProp}(\text{supp}(A))!$

3 if $(x : A) \vdash (c_A : C)$ and $(z, w : C) \vdash (c_\equiv : (z = w))$, for any $p : \text{supp}(A)$ we have $\text{match}(p, c_A, c_\equiv) : C$.

The hypotheses of the eliminator say exactly that $C$ is an h-prop and we have a map $A \to C$. 

\[\begin{array}{ccc}
A & \xrightarrow{\text{inhab}} & \text{supp}(A) \\
& \xleftarrow{c_A} & \\
& & C
\end{array}\]
The rest of logic

\[ \begin{align*}
P \text{ and } Q & \iff P \times Q \\
P \text{ implies } Q & \iff Q^P \\
\top \text{ (true)} & \iff 1 \\
\bot \text{ (false)} & \iff \emptyset \\
(\forall x : A)P(x) & \iff \prod_{x : A} B(x) \\
(\exists x : A)P(x) & \iff \text{supp}(\sum_{x : A} B(x))
\end{align*} \]

The magic of supp

Note: our ability to define “isProp” without using “supp” was crucial to our ability to define “supp” itself!

- Because we defined isProp using only paths, path-constructors can “universally force” a type to be an h-prop.
- Because isProp is an h-prop, these path-constructors have no other effect (give no extra data).
### Example

The **0-truncation** $\pi_0(A)$ has two constructors:

\[
(a : A) \vdash (\text{cpnt}(a) : \pi_0(A)) \\
(x, y : \pi_0(A)), (p, q : (x = y)) \vdash (\text{pp}(x, y, p, q) : (p = q))
\]

- The type of pp is precisely isHlevel(2, A).
- The eliminator says that $\pi_0(A)$ is a reflection of A into h-sets.

Now we can define

\[
\pi_1(A) := \pi_0(\Omega A)
\]

etc....

### Nonclassicality

#### Remark

h-sets and homotopy groups are a bit surprising.

1. A map $f : A \to B$ which induces $\pi_n(A) \sim \pi_n(B)$ for all $n : \mathbb{N}$ is **not** necessarily an equivalence!
   - Not closely related to non-CW-complex spaces.
   - It has to do with non-hypercomplete $(\infty, 1)$-toposes.
   - A reason not to call “equivalences” “weak equivalences”.

2. There may be types which do not admit a connected map from an h-set!
   - This happens in $\infty\text{Gpd}/X$ if $X$ is not discrete.
   - As a foundation, not every $\infty$-groupoid has an “underlying set” of objects (though it does have a $\pi_0$).
   - In particular, not every type has a cell decomposition.

These are “classicality properties” of $\infty\text{Gpd}$, like excluded middle and the axiom of choice in Set.
Localization

Given \( f : A \to B \).

**Definition**

- \( Z \) is **\( f \)**-local if \( Z^B \xrightarrow{\sim \circ f} Z^A \) is an equivalence.
- An **\( f \)**-localization of \( X \) is a reflection of \( X \) into \( f \)-local spaces.

**Examples**

- If \( f \) is \( S^n \to D^{n+1} \), then \( f \)-local means \((n-1)\)-truncated.
- Localization and completion at primes.
- Construction of \((\infty, 1)\)-toposes from \((\infty, 1)\)-presheaves.
- . . .

**h-isomorphisms**

Recall: \( f : A \to B \) is an **h-isomorphism** if we have

- A map \( g : B \to A \)
- A homotopy \( r : \prod_{a : A}(g(f(a)) = a) \)
- A map \( h : B \to A \)
- A homotopy \( s : \prod_{b : B}(f(g(b)) = b) \)

The type is\( \text{Hiso}(f) \) is an h-prop, equivalent to is\( \text{Equiv}(f) \).
Localization as a HIT

Definition
Given \( f : A \to B \) and \( X \), the localization \( L_f X \) has constructors:

\[
\begin{align*}
(x : X) \vdash (\text{tolocal}(x) : L_f X) \\
(g : A \to L_f X), (b : B) \vdash (\text{lsec}(g, b) : L_f X) \\
(g : A \to L_f X), (a : A) \vdash (\text{lsech}(g, a) : (\text{lsec}(g, f(a)) = g(a))) \\
(g : A \to L_f X), (b : B) \vdash (\text{lret}(g, b) : L_f X) \\
(h : B \to L_f X), (b : B) \vdash (\text{lreth}(h, b) : (\text{lret}(h \circ f, b) = h(b)))
\end{align*}
\]

The meaning of localization

- Of course, tolocal is a map \( X \to L_f X \).
- lsec is a map \((L_f X)^A \to (L_f X)^B\).
- lsech is a homotopy from \((L_f X)^A \xrightarrow{\text{lsec}} (L_f X)^B \xrightarrow{\circ f} (L_f X)^A\) to the identity.
- lret is a map \((L_f X)^A \to (L_f X)^B\).
- lreth is a homotopy from \((L_f X)^B \xrightarrow{\circ f} (L_f X)^A \xrightarrow{\text{lret}} (L_f X)^B\) to the identity.

Together, \((\text{lsec}, \text{lsech}, \text{lret}, \text{lreth})\) exactly inhabit “isHiso(− \circ f)”, i.e. “isLocal(\( f, X \)”).

Thus, \( L_f X \) is an \( f \)-localization of \( X \).
Recall:

- A model category has two weak factorization systems:
  
  (acyclic cofibrations, fibrations)
  (cofibrations, acyclic fibrations)

- Identity types correspond to the first WFS, using the mapping path space:

  \[ A \to [y : B, \, x : A, \, p : (g(x) = y)] \to B \]

- In topology, the second WFS is likewise related to the mapping cylinder.

  \[ A \to Mf \to B \]

Can we use HITs to construct this?

Acyclic fibrations

What is an acyclic fibration in type theory?

1. A fibration that is also an equivalence.
2. A fibration \( p : B \to A \) which admits a section \( s : A \to B \) (hence \( ps = 1_A \)) such that \( sp \sim 1_B \).
3. A dependent type \( B : A \to \text{Type} \) such that each \( B(a) \) is contractible.
What is a cofibration in type theory?

Actually, what is an acyclic cofibration in type theory? I.e. when does \( i: A \rightarrow B \) satisfy \( i \uplus p \) for any fibration \( p \)?

**Acyclic cofibrations**

**Theorem (Gambino-Garner)**

*If \( B \) is an inductive type and \( i \) is its only constructor, then \( i \uplus p \) for any fibration \( p \).*

\[
\begin{array}{ccc}
A & \xrightarrow{f} & Y \\
\downarrow^{i} & ? & \downarrow^{p} \\
B & \xrightarrow{g} & X
\end{array}
\]

**Proof.**

- \( p \) is a dependent type \( Y: X \rightarrow \text{Type} \); we want to define

\[
h: \prod_{b: B} Y(g(b))
\]

- By the eliminator, it suffices to specify \( h(b) \) when \( b = i(a) \).  
- But then we can take \( h(i(a)) := f(a) \). \( \Box \)
Path object factorizations

Example

refl: \( A \to \text{Id}_A \) is the only constructor of the identity type. Thus,

\[
A \xrightarrow{\text{refl}} \text{Id}_A \to A \times A
\]

is an (acyclic cofibration, fibration) factorization.

Some cofibrations

Theorem

If \( B \) is an inductive type and \( i: A \to B \) is one of its constructors, then \( i \square p \) for any acyclic fibration \( p \).

\[
\begin{array}{ccc}
A & \xrightarrow{f} & Y \\
\downarrow i & & \downarrow p \\
B & \xrightarrow{g} & X
\end{array}
\]

Proof.

- Now we have a section \( s: \prod_{x: x} Y(x) \).
- We define \( h: \prod_{b: B} Y(g(b)) \) with the eliminator of \( B \):
  - If \( b = i(a) \), take \( h(b) := f(a) \).
  - If \( b \) is some other constructor, take \( h(b) := s(g(b)) \).
Theorem

If $B$ is a higher inductive type and $i : A \to B$ is one of its point-constructors, then $i \Box p$ for any acyclic fibration $p$.

\[
\begin{array}{cccc}
A & f & Y \\
\downarrow i & ? & \downarrow p \\
B & g & X
\end{array}
\]

Proof.

- Now we have a section $s : \prod_{x : x} \mathcal{Y}(x)$.
- We define $h : \prod_{b : B} \mathcal{Y}(g(b))$ with the eliminator of $B$:
  - If $b = i(a)$, take $h(b) := f(a)$.
  - If $b$ is some other point-constructor, take $h(b) := s(g(b))$.
  - In the case of path-constructors, use the contractibility of the fibers of $p$.

The other factorization

Need a mapping cylinder for $f : A \to B$ that is dependent over $B$.

Definition

The mapping cylinder $Mf : B \to \text{Type}$ has three constructors:

\[
\begin{align*}
(b : B) \vdash \text{(right}(b) : Mf(b)) \\
(a : A) \vdash \text{(left}(a) : Mf(f(a))) \\
(a : A) \vdash \text{(glue}(a) : \text{(left}(a) = \text{right}(f(a))))
\end{align*}
\]

Theorem (Lumsdaine)

- This defines a WFS (cofibrations, acyclic fibrations).
- With the other WFS, and the type-theoretic equivalences, we have a model category (except for strict limits and colimits).
Conversely:

**Theorem (Lumsdaine–Shulman)**

A well-behaved combinatorial model category which models type theory as before (lccc etc.) also models all higher inductive types.

(In particular, simplicial sets.)

**Very rough sketch of proof.**

Combine the transfinite construction of initial algebras with the homotopy-theoretic small object argument.

---

**Elementary \((\infty, 1)\)-toposes**

**Proposal**

An elementary \((\infty, 1)\)-topos is an \((\infty, 1)\)-category \(C\) such that:

1. \(C\) has finite limits.
2. \(C\) is locally cartesian closed.
3. \(C\) has sufficiently many object classifiers.
4. \(C\) has sufficiently many “higher initial algebras”
   \((\Rightarrow C\) has finite colimits).  

**Conjecture**

Any elementary \((\infty, 1)\)-topos has an internal homotopy type theory modeling the univalence axiom and higher inductive types.