

# Issue IV: Higher Inductive Types

Maksym Sokhatskyi <sup>1</sup>

<sup>1</sup> National Technical University of Ukraine

Igor Sikorsky Kyiv Polytechnic Institute

May 4, 2019

## Abstract

CW-complexes are central to both homotopy theory and homotopy type theory (HoTT) and are encoded in cubical theorem-proving systems as higher inductive types (HIT), similar to recursive trees for (co)inductive types. We explore the basic primitives of homotopy theory, which are considered as a foundational basis in theorem-proving systems.

**Keywords:** Homotopy Theory, Type Theory

## Contents

<b>1</b>	<b>CW-Complexes</b>	<b>2</b>
1.1	Introduction: Countable Constructors . . . . .	3
1.2	Motivation: Higher Inductive Types . . . . .	3
1.3	Metatheory: Cohesive Topoi . . . . .	3
1.3.1	Geometric Proofs . . . . .	3
1.3.2	Flat Proofs . . . . .	3
1.3.3	Sharp Proofs . . . . .	3
1.3.4	Bose Proofs . . . . .	3
1.3.5	Fermi Proofs . . . . .	3
1.3.6	Linear Proofs . . . . .	3
<b>2</b>	<b>Higher Inductive Types</b>	<b>4</b>
2.1	Suspension . . . . .	5
2.2	Pushout . . . . .	6
2.3	Spheres . . . . .	7
2.4	Hub and Spokes . . . . .	8
2.5	Truncation . . . . .	9
2.6	Quotients . . . . .	10
2.7	Wedge . . . . .	11
2.8	Smash Product . . . . .	12
2.9	Join . . . . .	14
2.10	Colimit . . . . .	15
2.11	Coequalizers . . . . .	16

2.12 $K(G, n)$ . . . . .	18
2.13 Localization . . . . .	19

## 1 CW-Complexes

CW-complexes are spaces constructed by attaching cells of various dimensions. In HoTT, they are encoded as higher inductive types (HIT), where cells are constructors for points and paths.

**Definition 1.** (Cell Attachment). The attachment of an  $n$ -cell to a space  $X$  along  $f : S^{n-1} \rightarrow X$  is a pushout:

$$\begin{array}{ccc} S^{n-1} & \xrightarrow{f} & X \\ \downarrow \iota & & \downarrow j \\ D^n & \xrightarrow{g} & X \cup_f D^n \end{array}$$

Here,  $\iota : S^{n-1} \hookrightarrow D^n$  is the boundary inclusion, and  $X \cup_f D^n$  is the pushout that attaches an  $n$ -cell to  $X$  via  $f$ . The result depends on the homotopy class of  $f$ .

**Definition 2.** (CW-Complex). A CW-complex is a space  $X$ , constructed inductively by attaching cells, with a skeletal filtration:

- $(-1)$ -skeleton:  $X_{-1} = \emptyset$ .
- For  $n \geq 0$ , the  $n$ -skeleton  $X_n$  is obtained by attaching  $n$ -cells to  $X_{n-1}$ . For indices  $J_n$  and maps  $\{f_j : S^{n-1} \rightarrow X_{n-1}\}_{j \in J_n}$ ,  $X_n$  is the pushout:

$$\begin{array}{ccc} \coprod_{j \in J_n} S^{n-1} & \xrightarrow{\coprod f_j} & X_{n-1} \\ \downarrow \coprod \iota_j & & \downarrow i_n \\ \coprod_{j \in J_n} D^n & \xrightarrow{\coprod g_j} & X_n \end{array}$$

where  $\coprod_{j \in J_n} S^{n-1}$ ,  $\coprod_{j \in J_n} D^n$  are disjoint unions, and  $i_n : X_{n-1} \hookrightarrow X_n$  is the inclusion.

- $X$  is the colimit:

$$\emptyset = X_{-1} \hookrightarrow X_0 \hookrightarrow X_1 \hookrightarrow \dots \hookrightarrow X,$$

where  $X_n$  is the  $n$ -skeleton, and  $X = \text{colim}_{n \rightarrow \infty} X_n$ . The sequence is the skeletal filtration.

In HoTT, CW-complexes are higher inductive types (HIT) with constructors for cells and paths for attachment.

## 1.1 Introduction: Countable Constructors

Some HITs require an infinite number of constructors for spaces, such as Eilenberg-MacLane spaces or the infinite sphere  $S^\infty$ .

```
def S∞ : U
:= inductive { base
              | loop (n: ℕ) : base ≡ base
              }
```

Challenges include type checking, computation, and expressiveness.

Agda Cubical uses cubical primitives to handle HITs, supporting infinite constructors via HITs indexed by natural numbers, as colimits.

## 1.2 Motivation: Higher Inductive Types

HITs in HoTT enable direct encoding of topological spaces, such as CW-complexes. In homotopy theory, spaces are constructed by attaching cells via attaching maps. HoTT views types as spaces, elements as points, and equalities as paths, making HITs a natural choice. Standard inductive types cannot capture higher homotopies, but HITs allow constructors for points and paths. For example, the circle  $S^1$  (Definition 2) has a base point and a loop, encoding its fundamental group  $\mathbb{Z}$ . HITs avoid the use of multiple quotient spaces, preserving the synthetic nature of HoTT. In cubical type theory, paths are intervals (e.g.,  $\langle i \rangle$ ) with computational content, unlike propositional equalities, enabling efficient type checking in tools such as Agda Cubical.

## 1.3 Metatheory: Cohesive Topoi

### 1.3.1 Geometric Proofs

$$\mathfrak{R} \dashv \mathfrak{S} \dashv \&$$

For differential geometry, type theory incorporates primitive axioms of categorical meta-theoretical models of three Schreiber-Shulman functors: infinitesimal neighborhood ( $\mathfrak{S}$ ), reduced modality ( $\mathfrak{R}$ ), and infinitesimal discrete neighborhood ( $\&$ ).

### 1.3.2 Flat Proofs

### 1.3.3 Sharp Proofs

### 1.3.4 Bose Proofs

### 1.3.5 Fermi Proofs

### 1.3.6 Linear Proofs

$$\otimes \dashv x \dashv \multimap$$

For engineering applications (e.g., Milner’s  $\pi$ -calculus, quantum computing) and linear type theory, type theory embeds linear proofs based on the adjunction of the tensor and linear function spaces:  $(A \otimes B) \multimap A \simeq A \multimap (B \multimap C)$ , represented in a symmetric monoidal category  $\mathbf{D}$  for a functor  $[A, B]$  as:  $\mathbf{D}(A \otimes B, C) \simeq \mathbf{D}(A, [B, C])$ .

## 2 Higher Inductive Types

CW-complexes are central to HoTT and appear in cubical type checkers as HITs. Unlike inductive types (recursive trees), HITs encode CW-complexes, capturing points (0-cells) and higher paths (n-cells). The definition of an HIT specifies a CW-complex through cubical composition, an initial algebra in the cubical model.

## 2.1 Suspension

The suspension  $\Sigma A$  of a type  $A$  is a higher inductive type that constructs a new type by adding two points, called poles, and paths connecting each point of  $A$  to these poles. It is a fundamental construction in homotopy theory, often used to shift homotopy groups, e.g., obtaining  $S^{n+1}$  from  $S^n$ .

**Definition 3.** (Formation). For any type  $A : \mathcal{U}$ , there exists a suspension type  $\Sigma A : \mathcal{U}$ .

**Definition 4.** (Constructors). For a type  $A : \mathcal{U}$ , the suspension  $\Sigma A : \mathcal{U}$  is generated by the following higher inductive compositional structure:

$$\Sigma := \begin{cases} \text{north} \\ \text{south} \\ \text{merid} : (a : A) \rightarrow \text{north} \equiv \text{south} \end{cases}$$

```
def Σ (A: U) : U
:= inductive {
  | north
  | south
  | merid (a: A) : north ≡ south
}
```

**Theorem 1.** (Elimination). For a family of types  $B : \Sigma A \rightarrow \mathcal{U}$ , points  $n : B(\text{north})$ ,  $s : B(\text{south})$ , and a family of dependent paths

$$m : \Pi(a : A), \text{PathOver}(B, \text{merid}(a), n, s),$$

there exists a dependent map  $\text{Ind}_{\Sigma A} : (x : \Sigma A) \rightarrow B(x)$ , such that:

$$\begin{cases} \text{Ind}_{\Sigma A}(\text{north}) = n \\ \text{Ind}_{\Sigma A}(\text{south}) = s \\ \text{Ind}_{\Sigma A}(\text{merid}(a, i)) = m(a, i) \end{cases}$$

```
def PathOver (B: Σ A → U) (a: A) (n: B north) (s: B south) : U
:= PathP (λ i , B (merid a @ i)) n s
```

```
def Ind_ΣA (A: U) (B: Σ A → U) (n: B north) (s: B south)
(m: (a: A) → PathOver B (merid a) n s) : (x: Σ A) → B x
:= split { north → n | south → s | merid a @ i → m a @ i }
```

**Theorem 2.** (Computation).

$$\text{Ind}_{\Sigma A}(\text{north}) = n \text{Ind}_{\Sigma A}(\text{south}) = s \text{Ind}_{\Sigma A}(\text{merid}(a, i)) = m(a, i)$$

```
def Σ-β (A: U) (B: Σ A → U) (n: B north) (s: B south)
(m: (a: A) → PathOver B (merid a) n s) (x: Σ A)
: Path (B x) (Σ-I A B n s m x)
split { north → n | south → s | merid a @ i → m a @ i }
```

**Theorem 3.** (Uniqueness). Any two maps  $h_1, h_2 : (x : \Sigma A) \rightarrow B(x)$  are homotopic if they agree on north, south, and merid, i.e., if  $h_1(\text{north}) = h_2(\text{north})$ ,  $h_1(\text{south}) = h_2(\text{south})$ , and  $h_1(\text{merid } a) = h_2(\text{merid } a)$  for all  $a : A$ .

## 2.2 Pushout

The pushout (amalgamation) is a higher inductive type that constructs a type by gluing two types  $A$  and  $B$  along a common type  $C$  via maps  $f : C \rightarrow A$  and  $g : C \rightarrow B$ . It is a fundamental construction in homotopy theory, used to model cell attachment and cofibrant objects, generalizing the topological notion of a pushout.

**Definition 5.** (Formation). For types  $A, B, C : \mathcal{U}$  and maps  $f : C \rightarrow A$ ,  $g : C \rightarrow B$ , there exists a pushout  $\sqcup(A, B, C, f, g) : \mathcal{U}$ .

**Definition 6.** (Constructors). The pushout is generated by the following higher inductive compositional structure:

$$\sqcup := \begin{cases} \text{po}_1 : A \rightarrow \sqcup(A, B, C, f, g) \\ \text{po}_2 : B \rightarrow \sqcup(A, B, C, f, g) \\ \text{po}_3 : (c : C) \rightarrow \text{po}_1(f(c)) \equiv \text{po}_2(g(c)) \end{cases}$$

```
def  $\sqcup$  (A B C : U) (f : C  $\rightarrow$  A) (g : C  $\rightarrow$  B) : U
:= inductive {
  | po1 (a : A)
  | po2 (b : B)
  | po3 (c : C) : po1(f(c))  $\equiv$  po2(g(c))
}
```

**Theorem 4.** (Elimination). For a type  $D : \mathcal{U}$ , maps  $u : A \rightarrow D$ ,  $v : B \rightarrow D$ , and a family of paths  $p : (c : C) \rightarrow u(f(c)) \equiv v(g(c))$ , there exists a map  $\text{Ind}_{\sqcup} : \sqcup(A, B, C, f, g) \rightarrow D$ , such that:

$$\begin{cases} \text{Ind}_{\sqcup}(\text{po}_1(a)) = u(a) \\ \text{Ind}_{\sqcup}(\text{po}_2(b)) = v(b) \\ \text{Ind}_{\sqcup}(\text{po}_3(c, i)) = p(c, i) \end{cases}$$

```
def PathOver (A B C : U) (f : C  $\rightarrow$  A) (g : C  $\rightarrow$  B)
  (D :  $\sqcup$  A B C f g  $\rightarrow$  U)
  (c : C) (u : D (po1 (f c))) (v : D (po2 (g c))) : U
:= PathP ( $\lambda$  i, D (po3 c i)) u v

def Ind $\sqcup$  : (A B C : U) (f : C  $\rightarrow$  A) (g : C  $\rightarrow$  B)
  (D :  $\sqcup$  A B C f g  $\rightarrow$  U)
  (u : (a : A)  $\rightarrow$  D (po1 a))
  (v : (b : B)  $\rightarrow$  D (po2 b))
  (p : (c : C)  $\rightarrow$  PathOver D c (u (f c)) (v (g c)))
  : (x :  $\sqcup$  A B C f g)  $\rightarrow$  D x
:= split { po1 a  $\rightarrow$  u a | po2 b  $\rightarrow$  v b | po3 c @ i  $\rightarrow$  p c @ i }
```

**Theorem 5.** (Computation). For  $x : \sqcup(A, B, C, f, g)$ ,

$$\begin{cases} \text{Ind}_{\sqcup}(\text{po}_1(a)) \equiv u(a) \\ \text{Ind}_{\sqcup}(\text{po}_2(b)) \equiv v(b) \\ \text{Ind}_{\sqcup}(\text{po}_3(c, i)) \equiv p(c, i) \end{cases}$$

**Theorem 6.** (Uniqueness). Any two maps  $u, v : \sqcup(A, B, C, f, g) \rightarrow D$  are homotopic if they agree on  $\text{po}_1$ ,  $\text{po}_2$ , and  $\text{po}_3$ , i.e., if  $u(\text{po}_1(a)) = v(\text{po}_1(a))$  for all  $a : A$ ,  $u(\text{po}_2(b)) = v(\text{po}_2(b))$  for all  $b : B$ , and  $u(\text{po}_3(c)) = v(\text{po}_3(c))$  for all  $c : C$ .

**Example 1.** (Cell Attachment) The pushout models the attachment of an  $n$ -cell to a space  $X$ . Given  $f : S^{n-1} \rightarrow X$  and inclusion  $g : S^{n-1} \rightarrow D^n$ , the pushout  $\sqcup(X, D^n, S^{n-1}, f, g)$  is the space  $X \cup_f D^n$ , attaching an  $n$ -disk to  $X$  along  $f$ .

$$\begin{array}{ccc} S^{n-1} & \xrightarrow{f} & X \\ \downarrow g & & \downarrow \\ D^n & \longrightarrow & X \cup_f D^n \end{array}$$

## 2.3 Spheres

Spheres are higher inductive types with higher-dimensional paths, representing fundamental topological spaces.

**Definition 7.** (Pointed n-Spheres) The  $n$ -sphere  $S^n$  is defined recursively as a type in the universe  $\mathcal{U}$  using general recursion over dimensions:

$$S^n := \begin{cases} \text{point} : \mathbb{S}^n, \\ \text{surface} : \langle i_1, \dots, i_n \rangle [ (i_1 = 0) \rightarrow \text{point}, (i_1 = 1) \rightarrow \text{point}, \dots \\ (i_n = 0) \rightarrow \text{point}, (i_n = 1) \rightarrow \text{point} ] \end{cases}$$

**Definition 8.** (n-Spheres via Suspension) The  $n$ -sphere  $S^n$  is defined recursively as a type in the universe  $\mathcal{U}$  using general recursion over natural numbers  $\mathbb{N}$ . For each  $n \in \mathbb{N}$ , the type  $S^n : \mathcal{U}$  is defined as:

$$\mathbb{S}^n := \begin{cases} S^0 = \mathbf{2}, \\ S^{n+1} = \Sigma(S^n). \end{cases}$$

`def sphere :  $\mathbb{N} \rightarrow \mathcal{U} := \text{N-iter } \mathbf{U} \ \mathbf{2} \ \Sigma$`

This iterative definition applies the suspension functor  $\Sigma$  to the base type  $\mathbf{2}$  (0-sphere)  $n$  times to obtain  $S^n$ .

**Example 2.** (Sphere as CW-Complex) The  $n$ -sphere  $S^n$  can be constructed as a CW-complex with one 0-cell and one  $n$ -cell:

$$\begin{cases} X_0 = \{\text{base}\}, \text{ one point} \\ X_k = X_0 \text{ for } 0 < k < n, \text{ no additional cells} \\ X_n : \text{Attachment of an } n\text{-cell to } X_{n-1} = \{\text{base}\} \text{ along } f : S^{n-1} \rightarrow \{\text{base}\} \end{cases}$$

The constructor cell attaches the boundary of the  $n$ -cell to the base point, yielding the type  $S^n$ .

## 2.4 Hub and Spokes

The hub and spokes construction  $\odot$  defines an  $n$ -truncation, ensuring that the type has no non-trivial homotopy groups above dimension  $n$ . It models the type as a CW-complex with a hub (central point) and spokes (paths to points).

**Definition 9.** (Formation). For types  $S, A : \mathcal{U}$ , there exists a hub and spokes type  $\odot (S, A) : \mathcal{U}$ .

**Definition 10.** (Constructors). The hub and spokes type is freely generated by the following higher inductive compositional structure:

$$\odot := \begin{cases} \text{base} : A \rightarrow \odot (S, A) \\ \text{hub} : (S \rightarrow \odot (S, A)) \rightarrow \odot (S, A) \\ \text{spoke} : (f : S \rightarrow \odot (S, A)) \rightarrow (s : S) \rightarrow \text{hub}(f) \equiv f(s) \end{cases}$$

```
def  $\odot$  (S A: U) : U
:= inductive { base (x: A)
              | hub (f: S  $\rightarrow$   $\odot$  S A)
              | spoke (f: S  $\rightarrow$   $\odot$  S A) (s:S) : hub f  $\equiv$  f s
            }
```

**Theorem 7.** (Elimination). For a family of types  $P : \text{HubSpokes } S \ A \rightarrow \mathcal{U}$ , maps  $\text{pbase} : (x : A) \rightarrow P(\text{base } x)$ ,  $\text{phub} : (f : S \rightarrow \text{HubSpokes } S \ A) \rightarrow P(\text{hub } f)$ , and a family of paths  $\text{pspoke} : (f : S \rightarrow \text{HubSpokes } S \ A) \rightarrow (s : S) \rightarrow \text{PathP}(< i > P(\text{spoke } f \ s \ @ \ i)) (\text{phub } f) (P(f \ s))$ , there exists a map  $\text{hubSpokesInd} : (z : \text{HubSpokes } S \ A) \rightarrow P(z)$ , such that:

$$\begin{cases} \text{Ind}_{\odot} (\text{base } x) = \text{pbase } x \\ \text{Ind}_{\odot} (\text{hub } f) = \text{phub } f \\ \text{Ind}_{\odot} (\text{spoke } f \ s \ @ \ i) = \text{pspoke } f \ s \ @ \ i \end{cases}$$



## 2.5 Truncation

### Set Truncation

**Definition 11.** (Formation). Set truncation (0-truncation), denoted  $\|A\|_0$ , ensures that the type is a set, with homotopy groups vanishing above dimension 0.

**Definition 12.** (Constructors). For  $A : \mathcal{U}$ ,  $\|A\|_0 : \mathcal{U}$  is defined by the following higher inductive compositional structure:

$$\|-\|_0 := \begin{cases} \text{inc} : A \rightarrow \|A\|_0 \\ \text{squash} : (a, b : \|A\|_0) \rightarrow (p, q : a \equiv b) \rightarrow p \equiv q \end{cases}$$

```
def \|-\|_0 (A: U) : U
:= inductive { inc (a: A)
              | squash (a b: \|A\|_0) (p q: Path (\|A\|_0) a b)
                <i j> [ (i = 0) -> p @ j, (i = 1) -> q @ j,
                      (j = 0) -> a,      (j = 1) -> b ]
              }
```

**Theorem 8.** (Elimination  $\|A\|_0$ ) For a set  $B : \mathcal{U}$  (i.e.,  $\text{isSet}(B)$ ), and a map  $f : A \rightarrow B$ , there exists  $\text{setTruncRec} : \|A\|_0 \rightarrow B$ , such that  $\text{Ind}_{\|A\|_0}(\text{inc}(a)) = f(a)$ .

### Groupoid Truncation

**Definition 13.** (Formation). Groupoid truncation (1-truncation), denoted  $\|A\|_1$ , ensures that the type is a 1-groupoid, with homotopy groups vanishing above dimension 1.

**Definition 14.** (Constructors). For  $A : \mathcal{U}$ ,  $\|A\|_1 : \mathcal{U}$  is defined by the following higher inductive compositional structure:

$$\|-\|_1 := \begin{cases} \text{inc} : A \rightarrow \|A\|_1 \\ \text{squash} : (a, b : \|A\|_1) \rightarrow (p, q : a \equiv b) \rightarrow (r, s : p \equiv q) \rightarrow r \equiv s \end{cases}$$

```
def \|-\|_1 (A: U) : U
:= inductive { inc (a: A)
              | squash (a b: \|A\|_1) (p q: Path (\|A\|_1) a b)
                (r s: Path (Path (\|A\|_1) a b) p q) <i j k>
                [ (i = 0) -> r @ j @ k, (i = 1) -> s @ j @ k,
                  (j = 0) -> p @ k,      (j = 1) -> q @ k,
                  (k = 0) -> a,          (k = 1) -> b ]
              }
```

**Theorem 9.** (Elimination  $\|A\|_1$ ) For a 1-groupoid  $B : \mathcal{U}$  (i.e.,  $\text{isGroupoid}(B)$ ), and a map  $f : A \rightarrow B$ , there exists  $\text{Ind}_{\|A\|_1} : \|A\|_1 \rightarrow B$ , such that  $\text{Ind}_{\|A\|_1}(\text{inc}(a)) = f(a)$ .

## 2.6 Quotients

### Set Quotient Spaces

Quotient spaces are a powerful computational tool in type theory, embedded in the core of Lean.

**Definition 15.** (Formation). Set quotient spaces construct a type  $A$ , quotiented by a relation  $R : A \rightarrow A \rightarrow \mathcal{U}$ , ensuring that the result is a set.

**Definition 16.** (Constructors). For a type  $A : \mathcal{U}$  and a relation  $R : A \rightarrow A \rightarrow \mathcal{U}$ , the set quotient space  $A/R : \mathcal{U}$  is freely generated by the following higher inductive compositional structure:

$$A/R := \begin{cases} \text{quot} : A \rightarrow A/R \\ \text{ident} : (a, b : A) \rightarrow R(a, b) \rightarrow \text{quot}(a) \equiv \text{quot}(b) \\ \text{trunc} : (a, b : A/R) \rightarrow (p, q : a \equiv b) \rightarrow p \equiv q \end{cases}$$

```
def / (A : U) (R : A → A → U) : U
:= inductive { quot (a : A)
| ident (a b : A) (r : R a b) : quot(a) ≡ quot(b)
| trunc (a b : / A R) (p q : Path (/ A R) a b)
  <i j> [ (i = 0) → p @ j , (i = 1) → q @ j ,
        (j = 0) → a , (j = 1) → b ]
}
```

**Theorem 10.** (Elimination). For a family of types  $B : A/R \rightarrow \mathcal{U}$  with  $\text{isSet}(Bx)$ , and maps  $f : (x : A) \rightarrow B(\text{quot}(x))$ ,  $g : (a, b : A) \rightarrow (r : R(a, b)) \rightarrow \text{PathP}(< i > B(\text{ident}(a, b, r)) @ i)(f(a))(f(b))$ , there exists  $\text{Ind}_{A/R} : \Pi(x : A/R), B(x)$ , such that  $\text{Ind}_{A/R}(\text{quot}(a)) = f(a)$ .

### Groupoid Quotient Spaces

**Definition 17.** (Formation). Groupoid quotient spaces extend set quotient spaces to produce a 1-groupoid, including constructors for higher paths. Groupoid quotient spaces construct a type  $A$ , quotiented by a relation  $R : A \rightarrow A \rightarrow \mathcal{U}$ , ensuring that the result is a groupoid.

**Definition 18.** (Constructors). For a type  $A : \mathcal{U}$  and a relation  $R : A \rightarrow A \rightarrow \mathcal{U}$ , the groupoid quotient space  $A//R : \mathcal{U}$  includes constructors for points, paths, and higher paths, ensuring a 1-groupoid structure.

## 2.7 Wedge

The wedge of two pointed types  $A$  and  $B$ , denoted  $A \vee B$ , is a higher inductive type representing the union of  $A$  and  $B$  with identified base points. Topologically, it corresponds to  $A \times \{y_0\} \cup \{x_0\} \times B$ , where  $x_0$  and  $y_0$  are the base points of  $A$  and  $B$ , respectively.

**Definition 19.** (Formation). For pointed types  $A, B : \text{pointed}$ , the wedge  $A \vee B : \mathcal{U}$ .

**Definition 20.** (Constructors). The wedge is generated by the following higher inductive compositional structure:

$$\vee := \begin{cases} \text{winl} : A.1 \rightarrow A \vee B \\ \text{winr} : B.1 \rightarrow A \vee B \\ \text{wglue} : \text{winl}(A.2) \equiv \text{winr}(B.2) \end{cases}$$

```
def ∨ (A : pointed) (B : pointed) : U
:= inductive { winl (a : A.1)
              | winr (b : B.1)
              | wglue : winl(A.2) ≡ winr(B.2)
              }
```

**Theorem 11.** (Elimination). For a type  $P : A \vee B \rightarrow \mathcal{U}$ , maps  $f : A.1 \rightarrow C$ ,  $g : B.1 \rightarrow C$ , and a path  $p : \text{PathOverlue}(P, f(A.2), g(B.2))$ , there exists a map  $\text{Ind}_\vee : A \vee B \rightarrow C$ , such that:

$$\begin{cases} \text{Ind}(\text{winl}(a)) = f(a) \\ \text{Ind}(\text{winr}(b)) = g(b) \\ \text{Ind}(\text{wglue}(x)) = p(x) \end{cases}$$

```
def PathOverGlue : (P : A ∨ B → U)
  (p : P (inl (A.2))) (q : P (inr (B.2))) : U
:= PathP (λ i → P (wglue i)) p q

def Ind_∨ (A B : pointed) (C : U) (f : A.1 → C) (g : B.1 → C)
  (p : Path C (f A.2) (g B.2))
  : A ∨ B → C
:= split { winl a → f a | winr b → g b | wglue @ x → p @ x }
```

**Theorem 12.** (Computation). For  $z : \text{Wedge } AB$ ,

$$\begin{cases} \text{Ind}_\vee(\text{winl } a) \equiv f(a) \\ \text{Ind}_\vee(\text{winr } b) \equiv g(b) \\ \text{Ind}_\vee(\text{wglue } @ x) \equiv p @ x \end{cases}$$

**Theorem 13.** (Uniqueness). Any two maps  $h_1, h_2 : \text{Wedge } AB \rightarrow C$  are homotopic if they agree on  $\text{winl}$ ,  $\text{winr}$ , and  $\text{wglue}$ , i.e., if  $h_1(\text{winl } a) = h_2(\text{winl } a)$  for all  $a : A.1$ ,  $h_1(\text{winr } b) = h_2(\text{winr } b)$  for all  $b : B.1$ , and  $h_1(\text{wglue}) = h_2(\text{wglue})$ .

## 2.8 Smash Product

The smash product of two pointed types  $A$  and  $B$ , denoted  $A \wedge B$ , is a higher inductive type that quotients the product  $A \times B$  by the pushout  $A \sqcup B$ . It represents the space  $A \times B / (A \times \{y_0\} \cup \{x_0\} \times B)$ , collapsing the wedge to a single point.

**Definition 21.** (Formation). For pointed types  $A, B : \text{pointed}$ , the smash product  $A \wedge B : \mathcal{U}$ .

**Definition 22.** (Constructors). The smash product is generated by the following higher inductive compositional structure:

$$A \wedge B := \begin{cases} \text{basel} : A \wedge B \\ \text{baser} : A \wedge B \\ \text{proj}(x : A.1)(y : B.1) : A \wedge B \\ \text{gluel}(a : A.2) : \text{proj}(a, B.2) \equiv \text{basel} \\ \text{gluer}(b : B.2) : \text{proj}(A.2, b) \equiv \text{baser} \end{cases}$$

```
def  $\wedge$  (A : pointed) (B : pointed) : U
:= inductive {
  | basel
  | baser
  | proj (a : A.1) (b : B.1)
  | gluel (a : A.2) : proj(a, B.2)  $\equiv$  basel
  | gluer (a : B.2) : proj(A.2, b)  $\equiv$  baser
}
```

**Theorem 14.** (Elimination). For a family of types  $P : \text{Smash } A B \rightarrow \mathcal{U}$ , points  $\text{pbasel} : P(\text{basel})$ ,  $\text{pbaser} : P(\text{baser})$ , maps  $\text{pproj} : (x : A.1) \rightarrow (y : B.1) \rightarrow P(\text{proj } x y)$ , and a family of paths  $\text{pgluel} : (a : A.1) \rightarrow \text{pproj}(a, B.2) \equiv \text{pbasel}$ ,  $\text{pgluer} : (b : B.1) \rightarrow \text{pproj}(A.2, b) \equiv \text{pbaser}$ , there exists a map  $\text{Ind}_\wedge : (z : A \wedge B) \rightarrow P(z)$ , such that:

$$\begin{cases} \text{Ind}_\wedge(\text{basel}) = \text{pbasel} \\ \text{Ind}_\wedge(\text{baser}) = \text{pbaser} \\ \text{Ind}_\wedge(\text{proj } x y) = \text{pproj } x y \\ \text{Ind}_\wedge(\text{gluel } a @ i) = \text{pgluel } a @ i \\ \text{Ind}_\wedge(\text{gluer } b @ i) = \text{pgluer } b @ i \end{cases}$$

```
def Ind $\wedge$  (A B : pointed) (P : A  $\wedge$  B  $\rightarrow$  U)
  (pbasel : P basel) (pbaser : P baser)
  (pproj : (x : A.1)  $\rightarrow$  (y : B.1)  $\rightarrow$  P (proj x y))
  (pgluel : (a : A.1)  $\rightarrow$  PathP (<i> P (gluel a @ i)) (pproj a B.2) pbasel)
  (pgluer : (b : B.1)  $\rightarrow$  PathP (<i> P (gluer b @ i)) (pproj A.2 b) pbaser)
  : (z : A  $\wedge$  B)  $\rightarrow$  P z
:= split {
  | basel  $\rightarrow$  pbasel
  | baser  $\rightarrow$  pbaser
  | proj x y  $\rightarrow$  pproj x y
  | gluel a @ i  $\rightarrow$  pgluel a @ i
  | gluer b @ i  $\rightarrow$  pgluer b @ i
}
```

**Theorem 15.** (Computation). For a family of types  $P : A \wedge B \rightarrow \mathcal{U}$ , points  $\text{pbase} : P(\text{base})$ ,  $\text{pbaser} : P(\text{baser})$ , map  $\text{pproj} : (x : A.1) \rightarrow (y : B.1) \rightarrow P(\text{proj } x y)$ , and families of paths  $\text{pgluel} : (a : A.1) \rightarrow \text{PathP } (< i > P(\text{gluel } a @ i)) (\text{pproj } a B.2) \text{pbase}$ ,  $\text{pgluer} : (b : B.1) \rightarrow \text{PathP } (< i > P(\text{gluer } b @ i)) (\text{pproj } A.2 b) \text{pbaser}$ , the map  $\text{Ind}_\wedge : (z : A \wedge B) \rightarrow P(z)$  satisfies all equations for all variants of the predicate  $P$ :

$$\left\{ \begin{array}{l} \text{Ind}_\wedge (\text{base}) \equiv \text{pbase} \\ \text{Ind}_\wedge (\text{baser}) \equiv \text{pbaser} \\ \text{Ind}_\wedge (\text{proj } x y) \equiv \text{pproj } x y \\ \text{Ind}_\wedge (\text{gluel } a @ i) \equiv \text{pgluel } a @ i \\ \text{Ind}_\wedge (\text{gluer } b @ i) \equiv \text{pgluer } b @ i \end{array} \right.$$

**Theorem 16.** (Uniqueness). For a family of types  $P : A \wedge B \rightarrow \mathcal{U}$ , and maps  $h_1, h_2 : (z : A \wedge B) \rightarrow P(z)$ , if there exist paths  $e_{\text{base}} : h_1(\text{base}) \equiv h_2(\text{base})$ ,  $e_{\text{baser}} : h_1(\text{baser}) \equiv h_2(\text{baser})$ ,  $e_{\text{proj}} : (x : A.1) \rightarrow (y : B.1) \rightarrow h_1(\text{proj } x y) \equiv h_2(\text{proj } x y)$ ,  $e_{\text{gluel}} : (a : A.1) \rightarrow \text{PathP } (< i > h_1(\text{gluel } a @ i) \equiv h_2(\text{gluel } a @ i)) (e_{\text{proj } a B.2}) e_{\text{base}}$ ,  $e_{\text{gluer}} : (b : B.1) \rightarrow \text{PathP } (< i > h_1(\text{gluer } b @ i) \equiv h_2(\text{gluer } b @ i)) (e_{\text{proj } A.2 b}) e_{\text{baser}}$ , then  $h_1 \equiv h_2$ , i.e., there exists a path  $(z : A \wedge B) \rightarrow h_1(z) \equiv h_2(z)$ .

## 2.9 Join

The join of two types  $A$  and  $B$ , denoted  $A \vee B$ , is a higher inductive type that constructs a type by joining each point of  $A$  to each point of  $B$  via a path. Topologically, it corresponds to the join of spaces, forming a space that interpolates between  $A$  and  $B$ .

**Definition 23.** (Formation). For types  $A, B : \mathcal{U}$ , the join  $A * B : \mathcal{U}$ .

**Definition 24.** (Constructors). The join is generated by the following higher inductive compositional structure:

$$A \vee B := \begin{cases} \text{joinl} : A \rightarrow A \vee B \\ \text{joinr} : B \rightarrow A \vee B \\ \text{join}(a : A)(b : B) : \text{joinl}(a) \equiv \text{joinr}(b) \end{cases}$$

```
def ∨ (A : U) (B : U) : U
:= inductive { joinl (a : A)
              | joinr (b : B)
              | join (a : A) (b : B) : joinl(a) ≡ joinr(b)
            }
```

**Theorem 17.** (Elimination). For a type  $C : \mathcal{U}$ , maps  $f : A \rightarrow C$ ,  $g : B \rightarrow C$ , and a family of paths  $h : (a : A) \rightarrow (b : B) \rightarrow f(a) \equiv g(b)$ , there exists a map  $\text{Ind}_\vee : A \vee B \rightarrow C$ , such that:

$$\begin{cases} \text{Ind}_\vee(\text{joinl}(a)) = f(a) \\ \text{Ind}_\vee(\text{joinr}(b)) = g(b) \\ \text{Ind}_\vee(\text{join}(a, b, i)) = h(a, b, i) \end{cases}$$

```
def Ind_∨ (A B C : U) (f : A → C) (g : B → C)
  (h : (a : A) → (b : B) → Path C (f a) (g b))
  : A ∨ B → C
:= split { joinl a → f a
          | joinr b → g b
          | join a b @ i → h a b @ i
        }
```

**Theorem 18.** (Computation). For all  $z : A \vee B$ , and predicate  $P$ , the rules of  $\text{Ind}_\vee$  hold for all parameters of the predicate  $P$ .

**Theorem 19.** (Uniqueness). Any two maps  $h_1, h_2 : A \vee B \rightarrow C$  are homotopic if they agree on  $\text{joinl}$ ,  $\text{joinr}$ , and  $\text{join}$ .

## 2.10 Colimit

Colimits construct the limit of a sequence of types, connected by maps, e.g., propositional truncations.

**Definition 25.** (Colimit) For a sequence of types  $A : \text{nat} \rightarrow \mathcal{U}$  and maps  $f : (n : \mathbb{N}) \rightarrow A n \rightarrow A(\text{succ}(n))$ , the colimit type  $\text{colimit}(A, f) : \mathcal{U}$ .

$$\text{colim} := \begin{cases} \text{ix} : (n : \text{nat}) \rightarrow A n \rightarrow \text{colimit}(A, f) \\ \text{gx} : (n : \text{nat}) \rightarrow (a : A(n)) \rightarrow \text{ix}(\text{succ}(n), f(n, a)) \equiv \text{ix}(n, a) \end{cases}$$

```
def colimit (A : nat → U) (f : (n : nat) → A n → A (succ n)) : U
:= inductive { ix (n : nat) (x : A n)
              | gx (n : nat) (a : A n)
                <i> [ (i=0) → ix (succ n) (f n a),
                  (i=1) → ix n a ]
              }
```

**Theorem 20.** (Elimination colimit) For a type  $P : \text{colimit } Af \rightarrow \mathcal{U}$ , with  $p : (n : \text{nat}) \rightarrow (x : A n) \rightarrow P(\text{ix}(n, x))$  and  $q : (n : \text{nat}) \rightarrow (a : A n) \rightarrow \text{PathP}(\langle i \rangle P(\text{gx}(n, a) @ i))(p(\text{succ } n)(f n a))(p n a)$ , there exists  $i : \prod_{x : \text{colimit } Af} P(x)$ , such that  $i(\text{ix}(n, x)) = p n x$ .

## 2.11 Coequalizers

### Coequalizer

The coequalizer of two maps  $f, g : A \rightarrow B$  is a higher inductive type (HIT) that constructs a type consisting of elements in  $B$ , where  $f$  and  $g$  agree, along with paths ensuring this equality. It is a fundamental construction in homotopy theory, capturing the subspace of  $B$  where  $f(a) = g(a)$  for  $a : A$ .

**Definition 26.** (Formation). For types  $A, B : \mathcal{U}$  and maps  $f, g : A \rightarrow B$ , the coequalizer  $\text{coeq } ABfg : \mathcal{U}$ .

**Definition 27.** (Constructors). The coequalizer is generated by the following higher inductive compositional structure:

$$\text{Coeq} := \begin{cases} \text{inC} : B \rightarrow \text{Coeq}(A, B, f, g) \\ \text{glueC} : (a : A) \rightarrow \text{inC}(f(a)) \equiv \text{inC}(g(a)) \end{cases}$$

```
def Coeq (A B: U) (f g: A -> B) : U
:= inductive { inC (b: B)
              | glueC (a: A) : inC (f a) ≡ inC (g a)
            }
```

**Theorem 21.** (Elimination). For a type  $C : \mathcal{U}$ , map  $h : B \rightarrow C$ , and a family of paths  $y : (x : A) \rightarrow \text{Path}_C(h(fx), h(gx))$ , there exists a map  $\text{coequRec} : \text{coeq } ABfg \rightarrow C$ , such that:

$$\begin{cases} \text{coequRec}(\text{inC}(x)) = h(x) \\ \text{coequRec}(\text{glueC}(x, i)) = y(x, i) \end{cases}$$

```
def coequRec (A B C : U) (f g : A -> B) (h: B -> C)
  (y: (x : A) -> Path C (h (f x)) (h (g x)))
  : (z : coeq A B f g) -> C
:= split { inC x -> h x | glueC x @ i -> y x @ i }
```

**Theorem 22.** (Computation). For  $z : \text{coeq } ABfg$ ,

$$\begin{cases} \text{coequRec}(\text{inC } x) \equiv h(x) \\ \text{coequRec}(\text{glueC } x @ i) \equiv y(x) @ i \end{cases}$$

**Theorem 23.** (Uniqueness). Any two maps  $h_1, h_2 : \text{coeq } ABfg \rightarrow C$  are homotopic if they agree on  $\text{inC}$  and  $\text{glueC}$ , i.e., if  $h_1(\text{inC } x) = h_2(\text{inC } x)$  for all  $x : B$  and  $h_1(\text{glueC } a) = h_2(\text{glueC } a)$  for all  $a : A$ .

**Example 3.** (Coequalizer as Subspace) The coequalizer  $\text{coeq } ABfg$  represents the subspace of  $B$ , where  $f(a) = g(a)$ . For example, if  $A = B = \mathbb{R}$  and  $f(x) = x^2$ ,  $g(x) = x$ , the coequalizer captures the points where  $x^2 = x$ , i.e.,  $\{0, 1\}$ .



## Path Coequalizer

The path coequalizer is a higher inductive type that generalizes the coequalizer to handle pairs of paths in  $B$ . Given a map  $p : A \rightarrow (b_1, b_2 : B) \times (\text{Path}_B(b_1, b_2)) \times (\text{Path}_B(b_1, b_2))$ , it constructs a type where elements of  $A$  generate pairs of paths between points in  $B$ , with paths connecting the endpoints of these paths.

**Definition 28.** (Formation). For types  $A, B : \mathcal{U}$  and a map  $p : A \rightarrow (b_1, b_2 : B) \times (b_1 \equiv b_2) \times (b_1 \equiv b_2)$ , there exists a path coequalizer  $\text{Coeq}_{\equiv}(A, B, p) : \mathcal{U}$ .

**Definition 29.** (Constructors). The path coequalizer is generated by the following higher inductive compositional structure:

$$\text{Coeq}_{\equiv} := \begin{cases} \text{inP} : B \rightarrow \text{Coeq}_{\equiv}(A, B, p) \\ \text{glueP} : (a : A) \rightarrow \text{inP}(p(a).2.2.1 @ 0) \equiv \text{inP}(p(a).2.2.2 @ 1) \end{cases}$$

```
data Coeq≡ (A B : U) (p : A → Σ (b1 b2 : B), b1 ≡ b2 × b1 ≡ b2)
  = inP (b : B)
  | glueP (a : A) <i> [(i=0) → inP ((p a).2.2.1 @ 0),
                     (i=1) → inP ((p a).2.2.2 @ 1)]
```

**Theorem 24.** (Elimination). For a type  $C : \mathcal{U}$ , map  $h : B \rightarrow C$ , and a family of paths  $y : (a : A) \rightarrow h(p(a).2.2.1 @ 0) \equiv h(p(a).2.2.2 @ 1)$ , there exists a map  $\text{Ind-Coeq}_{\equiv} : \text{Coeq}_{\equiv}(A, B, p) \rightarrow C$ , such that:

$$\begin{cases} \text{coeqPRec}(\text{inP}(b)) = h(b) \\ \text{coeqPRec}(\text{glueP}(a, i)) = y(a, i) \end{cases}$$

```
def Ind-Coeq≡ (A B C : U)
  (p : A → Σ (b1 b2 : B) (x : Path B b1 b2), Path B b1 b2)
  (h : B → C) (y : (a : A) → Path C (h ((p a).2.2.1 @ 0)) (h ((p a).2.2.2 @ 1)))
  : (z : coeqP A B p) → C
:= split { inP b → h b | glueP a @ i → y a @ i }
```

**Theorem 25.** (Computation). For  $z : \text{coeqP } ABp$ ,

$$\begin{cases} \text{coeqPRec}(\text{inP } b) \equiv h(b) \\ \text{coeqPRec}(\text{glueP } a @ i) \equiv y(a) @ i \end{cases}$$

**Theorem 26.** (Uniqueness). Any two maps  $h_1, h_2 : \text{coeqP } ABp \rightarrow C$  are homotopic if they agree on  $\text{inP}$  and  $\text{glueP}$ , i.e., if  $h_1(\text{inP } b) = h_2(\text{inP } b)$  for all  $b : B$  and  $h_1(\text{glueP } a) = h_2(\text{glueP } a)$  for all  $a : A$ .

## 2.12 K(G,n)

Eilenberg-MacLane spaces  $K(G, n)$  have a single non-trivial homotopy group  $\pi_n(K(G, n)) = G$ . They are defined using truncations and suspensions.

**Definition 30.** ( $K(G, n)$ ) For an abelian group  $G : \text{abgroup}$ , the type  $KGn(G) : \text{nat} \rightarrow \mathcal{U}$ .

$$K(G, n) := \begin{cases} n = 0 \rightsquigarrow \text{discreteTopology}(G) \\ n \geq 1 \rightsquigarrow \|\Sigma^{n-1}(K1'(G.1, G.2.1))\|_n \end{cases}$$

```
def KGn (G: abgroup) : N -> U
:= split { zero -> discreteTopology G
          | succ n -> nTrunc (Σ (K1' (G.1, G.2.1)) n) (succ n)
          }
```

**Theorem 27.** (Elimination  $KGn$ ) For  $n \geq 1$ , a type  $B : \mathcal{U}$  with  $\text{isNGroupoid}(B, \text{succ } n)$ , and a map  $f : \text{suspension}(K1'G) \rightarrow B$ , there exists  $\text{rec}_{KGn} : KGnG(\text{succ } n) \rightarrow B$ , defined via  $\text{nTruncRec}$ .

### 2.13 Localization

Localization constructs an  $F$ -local type from a type  $X$ , with respect to a family of maps  $F_A : S(a) \rightarrow T(a)$ .

**Definition 31.** (Localization Modality) For a family of maps  $F_A : S(a) \rightarrow T(a)$ , the  $F$ -localization  $L_F^{AST}(X) : \mathcal{U}$ .

$$L_F^A(X) := \begin{cases} \text{center} : X \rightarrow L_{F_A}(X) \\ \text{ext}(a : A) \rightarrow (S(a) \rightarrow L_{F_A}(X)) : T(a) \rightarrow L_{F_A}(X) \\ \text{isExt}(a : A)(f : S(a) \rightarrow L_{F_A}(X)) \rightarrow (s : S(a)) : \text{ext}(a, f, F(a, s)) \equiv f(s) \\ \text{extEq}(a : A)(g, h : T(a) \rightarrow L_{F_A}(X)) \\ \quad (p : (s : S(a)) \rightarrow g(F(a, s)) \equiv h(F(a, s))) \\ \quad (t : T(a)) : g(t) \equiv h(t) \\ \text{isExtEq} : (a : A)(g, h : T(a) \rightarrow L_{F_A}(X)) \\ \quad (p : (s : S(a)) \rightarrow g(F(a, s)) \equiv h(F(a, s))) \\ \quad (s : S(a)) : \text{extEq}(a, g, h, p, F(a, s)) \equiv p(s) \end{cases}$$

```
data Localize (A X: U) (S T: A → U) (F : (x:A) → S x → T x)
= center (x: X)
| ext (a: A) (f: S a → Localize A X S T F) (t: T a)
| isExt (a: A) (f: S a → Localize A X S T F) (s: S a) <i>
  [ (i=0) → ext a f (F a s) , (i=1) → f s ]
| extEq (a: A) (g h: T a → Localize A X S T F)
  (p: (s : S a) → Path (Localize A X S T F) (g (F a s)) (h (F a s)))
  (t : T a) <i> [ (i=0) → g t , (i=1) → h t ]
| isExtEq (a: A) (g h : T a → Localize A X S T F)
  (p: (s : S a) → Path (T a → Localize A X S T F) (g (F a s)) (h (F a s)))
  (s : S a) <i> [ (i=0) → extEq a g h p (F a s) , (i=1) → p s ]
```

**Theorem 28.** (Localization Induction) For any  $P : \Pi_{X:U} L_{F_A}(X) \rightarrow U$  with  $\{n, r, s\}$ , satisfying coherence conditions, there exists  $i : \Pi_{x:L_{F_A}(X)} P(x)$ , such that  $i \cdot \text{center}_X = n$ .

### Conclusion

HITs directly encode CW-complexes in HoTT, bridging topology and type theory. They enable the analysis and manipulation of homotopical types.

## References

- [1] The Univalent Foundations Program, *Homotopy Type Theory: Univalent Foundations of Mathematics*, IAS, 2013.
- [2] C. Cohen, T. Coquand, S. Huber, A. Mörtberg, *Cubical Type Theory*, Journal of Automated Reasoning, 2018.
- [3] A. Mörtberg et al., *Agda Cubical Library*, <https://github.com/agda/cubical>, 2023.
- [4] M. Shulman, *Higher Inductive Types in HoTT*, <https://arxiv.org/abs/1705.07088>, 2017.
- [5] J. D. Christensen, M. Opie, E. Rijke, L. Scoccola, *Localization in Homotopy Type Theory*, <https://arxiv.org/pdf/1807.04155.pdf>, 2018.
- [6] E. Rijke, M. Shulman, B. Spitters, *Modalities in Homotopy Type Theory*, <https://arxiv.org/pdf/1706.07526v6.pdf>, 2017.
- [7] M. Riley, E. Finster, D. R. Licata, *Synthetic Spectra via a Monadic and Comonadic Modality*, <https://arxiv.org/pdf/2102.04099.pdf>, 2021.