# Issue IV: Higher Inductive Types

Maksym Sokhatskyi [1]

[1] National Technical University of Ukraine
Igor Sikorsky Kyiv Polytechnic Institute

May 4, 2019

**Анотація**

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

## Зміст

# 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 $f : S^{n-1} \to X$ is a pushout:

$$\begin{tikzcd} S^{n-1} \arrow[r, "f"] \arrow[d, "\iota"] & X \arrow[d, "j"] \\ D^n \arrow[r, "g"] & X \cup_f D^n \end{tikzcd}$$

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: $X_{-1} = \varnothing$.
- 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 $\{f_j : S^{n-1} \to X_{n-1}\}_{j \in J_n}$, $X_n$ is the pushout:

  $$\begin{tikzcd} \coprod_{j \in J_n} S^{n-1} \arrow[r, "\coprod f_j"] \arrow[d, "\coprod \iota_j"] & X_{n-1} \arrow[d, "i_n"] \\ \coprod_{j \in J_n} D^n \arrow[r, "\coprod g_j"] & X_n \end{tikzcd}$$

  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:

  $$\varnothing = X_{-1} \hookrightarrow X_0 \hookrightarrow X_1 \hookrightarrow \ldots \hookrightarrow X,$$

  where $X_n$ is the $n$-skeleton, and $X = \mathrm{colim}_{n \to \infty} X_n$. 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., $< i >$) 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

$$\Re \dashv \Im \dashv \&$$

For differential geometry, type theory incorporates primitive axioms of categorical meta-theoretical models of three Schreiber-Shulman functors: infinitesimal neighborhood ($\Im$), reduced modality ($\Re$), 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:

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

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

**Theorem 1.** (Elimination). For a family of types $B : \Sigma A \to \mathcal{U}$, 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 $\text{Ind}_{\Sigma A} : (x : \Sigma A) \to B(x)$, 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: ΣA -> 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 $h_1, h_2 : (x : \Sigma A) \to B(x)$ 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 $f : C \to A$ and $g : C \to B$. 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 $f : C \to A$, $g : C \to B$, 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:

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

```
def ⊔ (A B C: U) (f: C -> A) (g: C -> B) : U
 := inductive { po₁ (a: A)
              | po₂ (b: B)
              | po₃ (c: C) : po₁(f(c)) ≡ po₂(g(c))
              }
```

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

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

```
def PathOver (A B C : U) (f : C → A) (g : C → B)
     (D : ⊔ A B C f g → U)
     (c : C) (u : D (po₁ (f c))) (v : D (po₂ (g c))) : U
 := PathP (λ i, D (po₃ c i)) u v

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

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

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

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

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

$$\begin{tikzcd} S^{n-1} \arrow[r, "f"] \arrow[d, "g"] & X \arrow[d] \\ D^n \arrow[r] & X \cup_f D^n \end{tikzcd}$$

## 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:

$$\mathbb{S}^n := \begin{cases} \text{point} : \mathbb{S}^n, \\ \text{surface} : < i_1, ...i_n > [\ (i_1 = 0) \to \text{point}, (i_1 = 1) \to \text{point},\ ... \\ \qquad\qquad\qquad\qquad (i_n = 0) \to \text{point}, (i_n = 1) \to \text{point}\ ] \end{cases}$$

**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 : ℕ → U := ℕ-iter U 2 Σ
```

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:

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

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:

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

```
def ⊙ (S A: U) : U
 := inductive { base (x: A)
              | hub (f: S -> ⊙ S A)
              | spoke (f: S -> ⊙ S A) (s:S) : hub f ≡ f s
              }
```

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

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

```
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., isSet$(B)$), and a map $f : A \to B$, there exists setTruncRec : $\|A\|_0 \to B$, 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:

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

```
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., isGroupoid$(B)$), and a map $f : A \to B$, there exists $\text{Ind}_{\|A\|_1} : \|A\|_1 \to B$, 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 $R : A \to A \to \mathcal{U}$, ensuring that the result is a set.

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

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

```
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 $B : A/R \to \mathcal{U}$ with $\text{isSet}(Bx)$, and maps $f : (x : A) \to B(\text{quot}(x))$, $g : (a, b : A) \to (r : R(a, b)) \to \text{PathP}(< i > B(\text{ident}(a, b, r) \ @ \ i))(f(a))(f(b))$, 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 $R : A \to A \to \mathcal{U}$, ensuring that the result is a groupoid.

**Definition 18.** (Constructors). For a type $A : \mathcal{U}$ and a relation $R : A \to A \to \mathcal{U}$, 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 $A, B$ : pointed, the wedge $A \vee B : \mathcal{U}$.

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

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

```
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 $P : A \vee B\mathcal{U}$, maps $f : A.1 \to C$, $g : B.1 \to C$, and a path $p : \text{PathOverlue}(P, f(A.2), g(B.2))$, there exists a map $\text{Ind}_\vee : A \vee B \to C$, 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 $z$ : 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 $h_1, h_2$ : Wedge $AB \to C$ are homotopic if they agree on winl, winr, and 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 $A, B$ : 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 ∧ (A : pointed) (B : pointed) : U
 := inductive { basel
              | baser
              | proj (a : A.1) (b : B.1)
              | gluel (a : A.2) : proj(a,B.2) ≡ basel
              | gluer (a : B.2) : proj(A.2,b) ≡ baser
              }
```

**Theorem 14.** (Elimination). For a family of types $P$ : Smash $A\,B \to \mathcal{U}$, points pbasel : $P$(basel), pbaser : $P$(baser), maps pproj : $(x : A.1) \to (y : B.1) \to P\,(\text{proj}\,x\,y)$, and a family of paths pgluel : $(a : A.1) \to \text{pproj}(a, B.2) \equiv \text{pbasel}$, pgluer : $(b : B.1) \to \text{pproj}(A.2, b) \equiv \text{pbaser}$, there exists a map $\text{Ind}_\wedge : (z : A \wedge B) \to P(z)$, 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∧ (A B : pointed) (P : A ∧ B -> U)
    (pbasel : P basel) (pbaser : P baser)
    (pproj : (x : A.1) -> (y : B.1) -> P (proj x y))
    (pgluel : (a : A.1) -> PathP (<i> P (gluel a @ i)) (pproj a B.2) pbasel)
    (pgluer : (b : B.1) -> PathP (<i> P (gluer b @ i)) (pproj A.2 b) pbaser)
  : (z : A ∧ B) -> P z
 := split { basel -> pbasel | baser -> pbaser | proj x y -> pproj x y
          | gluel a @ i -> pgluel a @ i | gluer b @ i -> pgluer b @ i }
```

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

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

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

## 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:

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

```
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 $f : A \to C$, $g : B \to C$, and a family of paths $h : (a : A) \to (b : B) \to f(a) \equiv g(b)$, there exists a map $\text{Ind}_{\vee} : A \vee B \to C$, 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 $h_1, h_2 : A \vee B \to C$ are homotopic if they agree on joinl, joinr, and 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 $A : \text{nat} \to \mathcal{U}$ and maps $f : (n : \mathbb{N}) \to An \to A(\text{succ}(n))$, the colimit type $\text{colimit}(A, f) : \mathcal{U}$.

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

```
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 $P$ : colimit $Af \to \mathcal{U}$, with $p : (n : \text{nat}) \to (x : An) \to P(\text{ix}(n, x))$ and $q : (n : \text{nat}) \to (a : An) \to \text{PathP}(\langle i \rangle P(\text{gx}(n, a)@i))(p(\text{succ } n)(fna))(pna)$, there exists $i : \Pi_{x:\text{colimit } Af} P(x)$, such that $i(\text{ix}(n, x)) = pnx$.

## 2.11 Coequalizers

### Coequalizer

The coequalizer of two maps $f, g : A \to B$ 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 $f, g : A \to B$, the coequalizer coeq $ABfg : \mathcal{U}$.

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

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

```
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 $h : B \to C$, and a family of paths $y : (x : A) \to \text{Path}_C(h(fx), h(gx))$, there exists a map coequRec : coeq $ABfg \to C$, 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 $z :$ 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 $h_1, h_2 :$ coeq $ABfg \to C$ are homotopic if they agree on inC and 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 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 $p : A \to (b_1, b_2 : B) \times (\mathrm{Path}_B(b_1, b_2)) \times (\mathrm{Path}_B(b_1, b_2))$, 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 $p : A \to (b_1, b_2 : B) \times (b_1 \equiv b_2) \times (b_1 \equiv b_2)$, there exists a path coequalizer $\mathrm{Coeq}_{\equiv}(A, B, p) : \mathcal{U}$.

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

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

```
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 $h : B \to C$, and a family of paths $y : (a : A) \to h(p(a).2.2.1@0) \equiv h(p(a).2.2.2@1)$, there exists a map $\text{Ind-Coequ}_{\equiv} : \mathrm{Coeq}_{\equiv}(A, B, p) \to C$, such that:

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

```
def Ind-Coequ≡ (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 $z : \mathrm{coeqP}\ ABp$,

$$\begin{cases} \mathrm{coequPRec}(\mathrm{inP}\ b) \equiv h(b) \\ \mathrm{coequPRec}(\mathrm{glueP}\ a\,@\,i) \equiv y(a)\,@\,i \end{cases}$$

**Theorem 26.** (Uniqueness). Any two maps $h_1, h_2 : \mathrm{coeqP}\ ABp \to C$ are homotopic if they agree on inP and glueP, i.e., if $h_1(\mathrm{inP}\ b) = h_2(\mathrm{inP}\ b)$ for all $b : B$ and $h_1(\mathrm{glueP}\ a) = h_2(\mathrm{glueP}\ a)$ 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 $G$ : abgroup, the type $KGn(G)$ : nat $\to \mathcal{U}$.

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

```
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 isNGroupoid$(B, \text{succ } n)$, and a map $f$ : suspension$(K1'G) \to B$, there exists $\text{rec}_{KGn} : KGnG(\text{succ } n) \to B$, defined via nTruncRec.

## 2.13 Localization

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

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

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

```
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 $P : \Pi_{X:U} L_{F_A}(X) \to U$ 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.

# Література


[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.