Issue IV: Higher Inductive Types

Maxim Sokhatsky [1]

[1] National Technical University of Ukraine
Igor Sikorsky Kyiv Polytechnical Institute
April 22, 2025

Abstract

CW-complexes are key in homotopy type theory (HoTT) and are encoded in cubical type checkers as higher inductive types (HITs). Like recursive trees for (co)inductive types, HITs represent CW-complexes. An HIT defines a CW-complex using cubical composition as an initial algebra element in a cubical model. We explore HIT motivation, their topological role, and implementation in Agda Cubical, focusing on infinity constructors.
Keywords: Cellular Topology, Cubical Type Theory, HITs

Contents

1 CW-Complexes

CW-complexes are spaces built by attaching cells of increasing dimension. In HoTT, they are encoded as HITs, with cells as constructors for points and paths.

Definition 1. (Cell Attachment). Attaching 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, gluing 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 built inductively by attaching cells, with a skeletal filtration:

- The (-1)-skeleton is $X_{-1} = \varnothing$.
- 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 of:

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

  with X_n the n-skeleton, and $X = \mathrm{colim}_{n \to \infty} X_n$. The sequence is the skeletal filtration.

In HoTT, CW-complexes are HITs, with constructors for cells and path constructors for gluing.

Example 1. (Sphere as a CW-Complex). The n-sphere S^n is a CW-complex with one 0-cell and one n-cell:

- $X_0 = \{\text{base}\}$, a point.
- $X_k = X_0$ for $0 < k < n$, no cells added.
- X_n: Attach an n-cell to $X_{n-1} = \{\text{base}\}$ along $f : S^{n-1} \to \{\text{base}\}$: The cell constructor glues boundaries to base, yielding S^n.

1.1 Motivation for Higher Inductive Types

HITs in HoTT enable direct encoding of topological spaces like CW-complexes. In homotopy theory, spaces are built by gluing cells via attaching maps. HoTT views types as spaces, elements as points, and equalities as paths, making HITs a natural fit. 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 set-level quotients, preserving HoTT's synthetic nature. In cubical type theory, paths are intervals (e.g., $< i >$) with computational content, unlike propositional equalities, enabling efficient type checking in tools like Agda Cubical.

1.2 HITs with Infinity Constructors

Some HITs require infinite constructors for spaces like Eilenberg-MacLane spaces or the infinite sphere S^∞.

```
data SInf
   = base
   | loopn (n: Nat) <i> [ (i = 0) -> base, (i = 1) -> base ]
```

Challenges include type checking, computation, and expressivity.

Agda Cubical uses cubical primitives to handle HITs, supporting infinite constructors via indexed HITs.

```
data InfHIT (A: Type) : Type where
point : InfHIT A
pathn : (n: Nat) -> PathP (\lambda i -> InfHIT A) point point
```

2 Higher Inductive Types

CW-complexes are central in 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). Defining an HIT specifies a CW-complex via cubical composition, an initial algebra in a cubical model.

2.1 Suspension

The suspension ΣA of a type A is a higher inductive type (HIT) 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 type theory, often used to shift homotopy groups, e.g., producing S^{n+1} from S^n.

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

Definition 4. (Introduction) The suspension is generated by the following higher inductive composition structure:

$$\begin{cases} \text{north} : \Sigma A \\ \text{south} : \Sigma A \\ \text{merid} : (a : A) \to \text{Path}_{\Sigma A}(\text{north}, \text{south}) \end{cases}$$

```
data Suspension (A: U)
   = north
   | south
   | merid (a: A) <i> [ (i=0) -> north, (i=1) -> south ]
```

Theorem 1. (Elimination) For a type $B : \mathcal{U}$, points $n, s : B$, and a family of paths $m : (a : A) \to \text{Path}_B(n, s)$, there exists a map $\text{rec}_{\Sigma A} : \Sigma A \to B$ such that:

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

```
SuspensionRec (A B: U) (n s: B) (m: (a: A) -> Path B n s)
  : Suspension A -> B
  = split
    north -> n
    south -> s
    merid a @ i -> m a @ i
```

Theorem 2. (Computation) For $x : \Sigma A$,

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

```
SuspensionBeta (A B: U) (n s: B) (m: (a: A) -> Path B n s)
    (x: Suspension A)
  : Path B (SuspensionRec A B n s m x)
    (split north -> n; south -> s; merid a @ i -> m a @ i) x
  = idp B (SuspensionRec A B n s m x)
```

Theorem 3. (Uniqueness) Any two maps $h_1, h_2 : \Sigma A \to B$ 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 $a : A$.

Example 2. (Suspension of S^0) The suspension of the 0-sphere S^0 (two points) yields the 1-sphere S^1. If $A = S^0 = \{\text{base}_0, \text{base}_1\}$, then ΣS^0 has two points north, south, and two paths merid(base$_0$), merid(base$_1$), resembling the loop structure of S^1.

2.2 Pushout

The pushout is a higher inductive type (HIT) 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 type theory, used to model cell attachments and cofibrant objects, generalizing the topological notion of pushouts.

Definition 5. (Formation) For types $A, B, C : \mathcal{U}$ and maps $f : C \to A$, $g : C \to B$, the pushout pushout $ABCfg : \mathcal{U}$.

Definition 6. (Introduction) The pushout is generated by the following higher inductive composition structure:

$$\begin{cases} \text{po1} : A \to \text{pushout } ABCfg \\ \text{po2} : B \to \text{pushout } ABCfg \\ \text{po3} : (c : C) \to \text{Path}_{\text{pushout } ABCfg}(\text{po1 } (fc), \text{po2 } (gc)) \end{cases}$$

```
data pushout (A B C: U) (f: C -> A) (g: C -> B)
    = po1 (_: A)
    | po2 (_: B)
    | po3 (c: C) <i> [ (i = 0) -> po1 (f c) , (i = 1) -> po2 (g c) ]
```

Theorem 4. (Elimination) For a type $D : \mathcal{U}$, maps $h_A : A \to D$, $h_B : B \to D$, and a family of paths $h_C : (c : C) \to \text{Path}_D(h_A(fc), h_B(gc))$, there exists a map pushoutRec : pushout $ABCfg \to D$ such that:

$$\begin{cases} \text{pushoutRec}(\text{po1 } a) = h_A(a) \\ \text{pushoutRec}(\text{po2 } b) = h_B(b) \\ \text{pushoutRec}(\text{po3 } c \,@\, i) = h_C(c) \,@\, i \end{cases}$$

```
pushoutRec (A B C D: U) (f: C -> A) (g: C -> B)
           (hA: A -> D) (hB: B -> D) (hC: (c: C) -> Path D (hA (f c)) (hB (g c)))
  : pushout A B C f g -> D
  = split
    po1 a -> hA a
    po2 b -> hB b
    po3 c @ i -> hC c @ i
```

Theorem 5. (Computation) For $x : \text{pushout } ABCfg$,

$$\begin{cases} \text{pushoutRec}(\text{po1 } a) \equiv h_A(a) \\ \text{pushoutRec}(\text{po2 } b) \equiv h_B(b) \\ \text{pushoutRec}(\text{po3 } c \,@\, i) \equiv h_C(c) \,@\, i \end{cases}$$

Theorem 6. (Uniqueness) Any two maps $h_1, h_2 : \text{pushout } ABCfg \to D$ are homotopic if they agree on po1, po2, and po3, i.e., if $h_1(\text{po1 } a) = h_2(\text{po1 } a)$ for all $a : A$, $h_1(\text{po2 } b) = h_2(\text{po2 } b)$ for all $b : B$, and $h_1(\text{po3 } c) = h_2(\text{po3 } c)$ for all $c : C$.

Example 3. (Cell Attachment) The pushout models attaching an n-cell to a space X. Given $f : S^{n-1} \to X$ and the inclusion $g : S^{n-1} \to D^n$, the pushout pushout $XD^nS^{n-1}fg$ is the space $X \cup_f D^n$, gluing the 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 HITs with higher-dimensional paths, representing fundamental topological spaces like the circle (S^1) or 2-sphere (S^2).

Definition 7. (Spheres and Disks)

$$S^1 = \begin{cases} \text{base} \\ \text{loop } < i > \begin{cases} (i = 0) \mapsto \text{base} \\ (i = 1) \mapsto \text{base} \end{cases} \end{cases}$$

$$S^2 = \begin{cases} \text{point} \\ \text{surf} < i\, j > \begin{cases} (i = 0) \mapsto \text{point} \\ (i = 1) \mapsto \text{point} \\ (j = 0) \mapsto \text{point} \\ (j = 1) \mapsto \text{point} \end{cases} \end{cases}$$

```
data S1
  = base
  | loop <i> [ (i = 0) -> base, (i = 1) -> base ]
data S2
  = point
  | surf <i j> [ (i = 0) -> point, (i = 1) -> point,
                 (j = 0) -> point, (j = 1) -> point ]
```

Example 4. (Sphere as a CW-Complex) The n-sphere S^n is a CW-complex with one 0-cell and one n-cell:

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

```
data Sn (n: Nat)
  = base
  | cell <i1 ... in> [ (i1 = 0) -> base, (i1 = 1) -> base,
                       ...,
                       (in = 0) -> base, (in = 1) -> base ]
```

The cell constructor glues boundaries to base, yielding S^n.

2.4 Hub and Spoke

The Hub and Spoke construction defines n-truncations, ensuring a type has no non-trivial homotopy groups above dimension n. It models a type as a CW-complex with a hub (central point) and spokes (paths to points).

Definition 8. (Hub and Spokes) For types $S, A : \mathcal{U}$, the Hub and Spokes type hubSpokes $SA : \mathcal{U}$.

$$\begin{cases} \text{base} : A \to \text{hubSpokes } SA \\ \text{hub} : (S \to \text{hubSpokes } SA) \to \text{hubSpokes } SA \\ \text{spoke} : (f : S \to \text{hubSpokes } SA) \to (s : S) \to \text{Path}_{\text{hubSpokes } SA}(\text{hub } f, fs) \\ \text{hubEq} : (x, y : A) \to (p : S \to \text{Path}_A(x, y)) \to \text{Path}_{\text{hubSpokes } SA}(\text{base } x, \text{base } y) \\ \text{spokeEq} : (x, y : A) \to (p : S \to \text{Path}_A(x, y)) \to (s : S) \to \text{Path}_{\text{hubSpokes } SA}(\text{hubEq } xyp, \text{base } (ps)) \end{cases}$$

```
data hubSpokes (S A: U)
   = base (x: A)
   | hub (f: S -> hubSpokes S A)
   | spoke (f: S -> hubSpokes S A) (s:S)
     <i> [ (i=0) -> hub f , (i=1) -> f s ]
   | hubEq (x y: A) (p: S -> Path A x y)
     <i> [ (i=0) -> base x , (i=1) -> base y ]
   | spokeEq (x y: A) (p: S -> Path A x y) (s: S)
     <i> [ (i=0) -> hubEq x y p , (i=1) -> base (p s) ]
```

Theorem 7. (Elimination hubSpokes) For a type $B : \mathcal{U}$, a map $g : A \to B$, a point $h : (S \to \text{hubSpokes } SA) \to B$, and path maps ensuring coherence, there exists $\text{rec}_{\text{hubSpokes}} : \text{hubSpokes } SA \to B$, such that $\text{rec}_{\text{hubSpokes}}(\text{base } x) = g(x)$ and $\text{rec}_{\text{hubSpokes}}(\text{hub } f) = h(f)$.

2.5 Set-Truncations

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

Definition 9. (Set Truncation) For $A : \mathcal{U}$, $\|A\|_0 : \mathcal{U}$.

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

```
data setTrunc (A: U)
   = inc (a: A)
   | squash (a b: setTrunc A) (p q: Path (setTrunc A) 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., $\text{isSet}(B)$), a map $f : A \to B$, there exists $\text{setTruncRec} : \|A\|_0 \to B$, such that $\text{setTruncRec}(\text{inc}(a)) = f(a)$.

2.6 Groupoid-Truncations

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

Definition 10. (Groupoid Truncation) For $A : \mathcal{U}$, $\|A\|_1 : \mathcal{U}$.

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

```
data grpdTrunc (A: U)
   = inc (a: A)
   | squash (a b: grpdTrunc A)
            (p q: Path (grpdTrunc A) a b)
            (r s: Path (Path (grpdTrunc A) 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., $\text{isGroupoid}(B)$), a map $f : A \to B$, there exists $\text{grpdTruncRec} : \|A\|_1 \to B$, such that $\text{grpdTruncRec}(\text{inc}(a)) = f(a)$.

2.7 Set-Quotients

Set quotients construct a type A quotiented by a relation $R : A \to A \to \mathcal{U}$, ensuring the result is a set.

Definition 11. (Set Quotient) For a type $A : \mathcal{U}$ and a relation $R : A \to A \to \mathcal{U}$, the set quotient setQuot $AR : \mathcal{U}$.

$$\begin{cases} \text{quotient} : A \to \text{setQuot } AR \\ \text{identification} : (a, b : A) \to Rab \to \text{Path}_{\text{setQuot } AR}(\text{quotient } a, \text{quotient } b) \\ \text{trunc} : (a, b : \text{setQuot } AR) \to (p, q : \text{Path}_{\text{setQuot } AR}(a, b)) \to \text{Path}_{\text{Path}_{\text{setQuot } AR}(a,b)}(p, q) \end{cases}$$

```
data setQuot (A: U) (R: A -> A -> U)
   = quotient (a: A)
   | identification (a b: A) (r: R a b)
     <i> [ (i=0) -> quotient a, (i=1) -> quotient b ]
   | trunc (a b : setQuot A R) (p q : Path (setQuot A R) a b)
     <i j> [ (i = 0) -> p @ j , (i = 1) -> q @ j ,
             (j = 0) -> a ,     (j = 1) -> b ]
```

Theorem 10. (Elimination setQuot) For a type family B : setQuot $AR \to \mathcal{U}$ with isSet(Bx), and maps $f : (x : A) \to B(\text{quotient } x)$, $g : (a, b : A) \to (r : Rab) \to \text{PathP}(\langle i \rangle B(\text{identification } abr@i))(fa)(fb)$, there exists setQuotElim : $\Pi_{x:\text{setQuot } AR} B(x)$, such that setQuotElim(quotient a) $= fa$.

2.8 Groupoid-Quotients

Groupoid quotients extend set quotients to produce a 1-groupoid, incorporating higher path constructors.

Definition 12. (Groupoid Quotient) For a type $A : \mathcal{U}$ and a relation $R : A \to A \to \mathcal{U}$, the groupoid quotient grpdQuot $AR : \mathcal{U}$ includes constructors for points, paths, and higher paths ensuring 1-groupoid structure. (Note: Full definition requires additional structure, partially omitted for brevity.)

2.9 Colimits

Colimits construct the limit of a sequence of types connected by maps, such as propositional truncations.

Definition 13. (Colimit) For a sequence of types $A : \text{nat} \to \mathcal{U}$ and maps $f : (n : \text{nat}) \to An \to A(\text{succ } n)$, the colimit type colimit $Af : \mathcal{U}$.

$$\begin{cases} \text{ix} : (n : \text{nat}) \to An \to \text{colimit } Af \\ \text{gx} : (n : \text{nat}) \to (a : An) \to \text{Path}_{\text{colimit } Af}(\text{ix}(\text{succ } n)(fna), \text{ix}na) \end{cases}$$

```
data colimit (A : nat -> U)
             (f : (n : nat) -> A n -> A (succ n))
   = 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 11. (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 $i(\text{ix}(n,x)) = pnx$.

2.10 Equalizer

The equalizer 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 enforcing this equality. It is a fundamental construction in homotopy type theory, capturing the subspace of B where $f(a) = g(a)$ for $a : A$.

Definition 14. (Formation) For types $A, B : \mathcal{U}$ and maps $f, g : A \to B$, the equalizer $\text{coeq } ABfg : \mathcal{U}$.

Definition 15. (Introduction) The equalizer is generated by the following higher inductive composition structure:

$$\begin{cases} \text{inC} : B \to \text{coeq } ABfg \\ \text{glueC} : (a : A) \to \text{Path}_{\text{coeq } ABfg}(\text{inC } (fa), \text{inC } (ga)) \end{cases}$$

```
data coeq (A B: U) (f g: A -> B)
   = inC (_: B)
   | glueC (a: A) <i> [(i=0) -> inC (f a), (i=1) -> inC (g a) ]
```

Theorem 12. (Elimination) For a type $C : \mathcal{U}$, a 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 $\text{coequRec} : \text{coeq } ABfg \to C$ such that:

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

```
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 13. (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 14. (Uniqueness) Any two maps $h_1, h_2 : \text{coeq } ABfg \to C$ 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 5. (Equalizer as Subspace) The equalizer $\text{coeq } ABfg$ 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 equalizer captures points where $x^2 = x$, i.e., $\{0, 1\}$.

2.11 Path-Equalizer

The path-equalizer is a higher inductive type that generalizes the equalizer 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 induce pairs of paths between points in B, with paths connecting the endpoints of these paths.

Definition 16. (Formation) For types $A, B : \mathcal{U}$ and a map $p : A \to (b_1, b_2 : B) \times (\mathrm{Path}_B(b_1, b_2)) \times (\mathrm{Path}_B(b_1, b_2))$, the path-equalizer coeqP $ABp : \mathcal{U}$.

Definition 17. (Introduction) The path-equalizer is generated by the following higher inductive composition structure:

$$\begin{cases} \mathrm{inP} : B \to \mathrm{coeqP}\ ABp \\ \mathrm{glueP} : (a : A) \to \mathrm{Path}_{\mathrm{coeqP}\ ABp}(\mathrm{inP}\ (((p\,a).2.2.1) \,@\, 0), \mathrm{inP}\ (((p\,a).2.2.2) \,@\, 1)) \end{cases}$$

```
data coeqP (A B: U) (p : A -> (b1 b2: B) * (_: Path B b1 b2) * (Path B 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 15. (Elimination) For a type $C : \mathcal{U}$, a map $h : B \to C$, and a family of paths $y : (a : A) \to \mathrm{Path}_C(h(((p\,a).2.2.1) \,@\, 0), h(((p\,a).2.2.2) \,@\, 1))$, there exists a map coequPRec : coeqP $ABp \to C$ such that:

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

```
coequPRec (A B C : U) (p : A -> (b1 b2: B) * (_: 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) @
  : (z : coeqP A B p) -> C
  = split
    inP b -> h b
    glueP a @ i -> y a @ i
```

Theorem 16. (Computation) For z : 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 17. (Uniqueness) Any two maps h_1, h_2 : 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 $a : A$.

Example 6. (Path-Equalizer for Homotopy) The path-equalizer can model spaces where elements of A specify pairs of paths between points in B. For instance, if $p(a)$ provides two paths from b_1 to b_2 in B, coeqP constructs a type connecting the starting and ending points of these paths, useful in studying homotopy classes.

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 18. (K(G,n)) For an abelian group G : abgroup, the type $KGnG$: nat $\rightarrow \mathcal{U}$.

$$\begin{cases} n = 0 : KGnG0 = \text{discreteTopology}(G) \\ n \geq 1 : KGnG(\text{succ } n) = \text{nTrunc}(\text{suspension}(K1'(G.1, G.2.1))n)(\text{succ } n) \end{cases}$$

```
KGn (G: abgroup)
  : nat -> U
  = split
    zero -> discreteTopology G
    succ n -> nTrunc (suspension (K1' (G.1,G.2.1)) n) (succ n)
```

Theorem 18. (Elimination KGn) For $n \geq 1$, a type $B : \mathcal{U}$ with isNGroupoid$(B, \text{succ } n)$, and a map f : suspension$(K1'G) \rightarrow B$, there exists $\text{rec}_{KGn} : KGnG(\text{succ } n) \rightarrow B$, 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 19. (Localization Modality) For a family of maps $F_A : S(a) \to T(a)$, the F-localization $L_F^{AST}(X) : \mathcal{U}$.

$$\begin{cases} \text{center} : X \to L_{F_A}(X) \\ \text{ext} : (a : A) \to (S(a) \to L_{F_A}(X)) \to T(a) \to L_{F_A}(X) \\ \text{isExt} : (a : A) \to (f : S(a) \to L_{F_A}(X)) \to (s : S(a)) \to \text{Path}_{L_{F_A}(X)}(\text{ext } af(Fas), fs) \\ \text{extEq} : (a : A) \to (g, h : T(a) \to L_{F_A}(X)) \to (p : (s : S(a)) \to \text{Path}_{L_{F_A}(X)}(g(Fas), h(Fas))) \to (t : T(a)) \to \\ \text{isExtEq} : (a : A) \to (g, h : T(a) \to L_{F_A}(X)) \to (p : (s : S(a)) \to \text{Path}_{L_{F_A}(X)}(g(Fas), h(Fas))) \to (s : S(a)) \to \end{cases}$$

```
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 19. (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 $i \cdot \text{center}_X = n$.

3 Conclusion

HITs encode CW-complexes in HoTT, bridging topology and type theory. They capture cell attachments, with examples like spheres, tori, and truncations. Infinity constructors extend HITs to infinite spaces, handled by Agda Cubical's primitives and indexed HITs.

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.