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 fundamental objects in homotopy type theory (HoTT) and are encoded in cubical type checkers as higher inductive types (HITs). Just as regular (co)inductive types can be described as recursive terminating or non-terminating trees, HITs can be viewed as CW-complexes. Defining an HIT involves specifying a CW-complex directly using cubical homogeneous composition structures as elements of an initial algebra in a cubical model. This article explores the motivation for HITs, their role in encoding topological spaces, and their implementation in libraries like Agda Cubical, with a focus on handling infinity constructors.

Keywords: Cellular Piecewise Topology, Cubical Type Theory, Higher Inductive Types

Contents

1 Higher Inductive Types

CW-complexes are fundamental in homotopy type theory and are implemented in cubical type checkers as higher inductive types (HITs). Unlike regular inductive types, which can be seen as recursive trees (terminating for inductive types, non-terminating for coinductive types), HITs generalize this to encode CW-complexes, capturing both points (0-cells) and higher-dimensional paths or homotopies (n-cells). Defining an HIT means specifying a CW-complex using cubical homogeneous composition structures, which serve as elements of an initial algebra in a cubical model.

Definition 1. (Pushout). Pushouts are a key example of HITs, used to formally define cell attachments and other cofibrant objects.

```
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) ]
```

Definition 2. (Spheres and Disks). Spheres are classic examples of HITs, defined using paths in higher dimensions.

```
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 ]
```

2 CW-Complexes

In homotopy theory, CW-complexes are spaces constructed by iteratively attaching cells of increasing dimension. In HoTT, these are naturally encoded as higher inductive types, where cells correspond to constructors for points, paths, and higher homotopies.

Definition 3. (Cell Attachment). Attaching an n-cell to a space X along a map $f : S^{n-1} \to X$ is defined as a pushout in the category of spaces:

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

where $\iota : S^{n-1} \hookrightarrow D^n$ is the inclusion of the $(n-1)$-sphere as the boundary of the n-disk, and $X \cup_f D^n$ is the pushout, representing the space obtained by gluing an n-cell to X via f. The homotopy type of $X \cup_f D^n$ depends on the homotopy class of f.

Definition 4. (CW-Complex). A CW-complex is a space X constructed inductively by attaching cells of increasing dimension, equipped with a skeletal filtration. Formally:

- The (-1)-skeleton is the empty space, $X_{-1} = \varnothing$.
- For each $n \geq 0$, the n-skeleton X_n is a space obtained by attaching a (possibly infinite) collection of n-cells to the $(n-1)$-skeleton X_{n-1}. For a set of indices J_n and attaching maps $\{f_j : S^{n-1} \to X_{n-1}\}_{j \in J_n}$, the n-skeleton 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 $\coprod_{j \in J_n} S^{n-1}$ and $\coprod_{j \in J_n} D^n$ denote disjoint unions over the index set J_n, and $i_n : X_{n-1} \hookrightarrow X_n$ is the inclusion of the $(n-1)$-skeleton into the n-skeleton.

- A CW-complex X is the colimit of the sequence of inclusions:

$$\varnothing = X_{-1} \hookrightarrow X_0 \hookrightarrow X_1 \hookrightarrow X_2 \hookrightarrow \ldots \hookrightarrow X,$$

  where each X_n is the n-skeleton, and $X = \mathrm{colim}_{n \to \infty} X_n$. The sequence $\{X_n\}_{n \geq -1}$ is called the skeletal filtration of X.

In HoTT, this construction is encoded as a higher inductive type, where constructors correspond to cells and their boundaries, with path constructors specifying the gluing via attaching maps.

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

- $X_0 = \{\text{base}\}$, a single point.
- $X_k = X_0$ for $0 < k < n$, as no cells are attached in dimensions 1 to $n-1$.
- X_n is obtained by attaching an n-cell to $X_{n-1} = \{\text{base}\}$ along the constant map $f : S^{n-1} \to \{\text{base}\}$. In cubical syntax:

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

  Here, cell is an n-dimensional constructor gluing the boundaries to base, yielding S^n.

3 Motivation for Higher Inductive Types

The motivation for HITs in HoTT stems from the need to represent topological spaces, such as CW-complexes, directly within type theory. In classical homotopy theory, spaces are constructed by gluing cells (points, paths, surfaces, etc.) via attaching maps. HoTT interprets types as spaces, elements as points, and equalities as paths, making it natural to encode spaces as types. However, standard inductive types are insufficient for higher-dimensional structures, as they cannot represent paths between paths (homotopies) or higher homotopies.

HITs extend inductive types by allowing constructors for both points and higher-dimensional paths. For example, the circle S^1 (Definition 1) is defined not only by a base point but also by a loop connecting the base point to itself, capturing its fundamental group $\mathbb{Z}$. This direct encoding avoids the need for set-level constructions or quotient types, preserving the synthetic nature of HoTT.

HITs also enable computational tractability in cubical type theory. Unlike Martin-Löf Type Theory, where path equality is propositional, cubical type theory treats paths as intervals (e.g., $< i >$) with computational content, allowing type checkers to verify equalities in HITs efficiently. This makes HITs practical for formalizing algebraic topology in proof assistants like Agda Cubical.

3.1 Examples of Higher Inductive Types

Beyond spheres, HITs can encode more complex spaces:

- **Torus**: The torus T^2 is a 2-dimensional CW-complex with one 0-cell, two 1-cells (loops), and one 2-cell gluing them.

  ```
  data Torus
     = point
     | loop1 <i> [ (i = 0) -> point , (i = 1) -> point ]
     | loop2 <i> [ (i = 0) -> point , (i = 1) -> point ]
     | square <i j> [ (i = 0) -> loop1 @ j , (i = 1) -> loop1 @ j ,
                      (j = 0) -> loop2 @ i , (j = 1) -> loop2 @ i ]
  ```

  Here, square ensures the two loops commute, encoding the torus's fundamental group $\mathbb{Z} \times \mathbb{Z}$.

- **Suspension**: The suspension ΣA of a type A adds two points (north and south poles) and paths from each point of A to both poles.

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

  For $A = S^0$, this yields S^1, illustrating how HITs build higher spheres.

- **Truncation**: The n-truncation $\|A\|_n$ of a type A collapses higher homotopies above dimension n.

```
data Trunc (n: Nat) (A: U)
  = hub (a: A)
  | spoke (p: Path A a b) <i> [ (i = 0) -> hub a, (i = 1) -> hub b ]
```

This is an HIT with paths ensuring connectivity up to dimension n.

These examples highlight how HITs encode topological spaces by specifying points, paths, and higher homotopies, mirroring the cell structure of CW-complexes.

4 HITs with Infinity Constructors

Some HITs require infinitely many constructors to encode spaces with infinite cell structures, such as Eilenberg-MacLane spaces $K(G, n)$ or infinite-dimensional spheres. For instance, consider an infinite-dimensional type like the infinite sphere S^∞, which can be approximated as a colimit of finite spheres:

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

Here, loopn introduces a path for each natural number n, representing an infinite family of 1-cells. Such HITs pose challenges:

1. **Type Checking**: Infinite constructors complicate termination and coverage checking in type checkers, as the induction principle must account for all constructors. 2. **Computation**: Cubical type theory must ensure that compositions involving infinite paths are well-defined and computable. 3. **Expressivity**: Infinity constructors are needed for spaces with non-trivial higher homotopy groups, but they risk making definitions non-constructive.

4.1 Agda Cubical and Infinity Constructors

The Agda Cubical library, built on cubical type theory, addresses these challenges by leveraging cubical primitives and HITs. Agda Cubical provides:

- **Cubical Primitives**: Intervals $I = [0, 1]$, path types PathP, and composition operations (hcomp, glue) allow HITs to be defined with computational content. For example, the S^1 definition in Agda Cubical uses:

```
data S1 : Type where
  base : S1
  loop : PathP (\lambda i -> S1) base base
```

This extends to higher dimensions and infinite constructors by parameterizing over indices (e.g., Nat).

- **HIT Support**: Agda Cubical supports HITs natively via cubical syntax, allowing definitions like the torus or suspension. For infinite constructors, Agda uses indexed HITs, where constructors are parameterized by a type (e.g., Nat) to emulate infinite families:

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

- **Normalization**: Agda Cubical ensures that HITs with infinite constructors normalize correctly by reducing compositions to canonical forms, leveraging the cubical model's coherence properties.

- **Limitations**: Infinite constructors can lead to performance issues in type checking, as Agda must verify coverage for all cases. Agda Cubical mitigates this by using strict evaluation for indices and optimizing composition algorithms.

For example, the Eilenberg-MacLane space $K(\mathbb{Z}, 1)$, which is homotopy equivalent to S^1, can be defined with an infinite family of paths to capture its loop structure, but Agda Cubical often uses finite approximations or coinductive techniques to manage complexity.

5 Conclusion

Higher inductive types provide a powerful framework for encoding CW-complexes in HoTT, bridging topology and type theory. By allowing constructors for points, paths, and higher homotopies, HITs capture the iterative cell attachment process of CW-complexes, as seen in the formal definition of skeletal filtrations and pushout constructions. Examples like spheres, tori, and suspensions illustrate how HITs mirror topological constructions, while infinity constructors extend this to infinite-dimensional spaces. Agda Cubical's cubical primitives and indexed HITs make these constructions computationally tractable, despite challenges with infinite constructors. As cubical type theory advances, HITs will continue to enable synthetic homotopy theory, formalizing complex topological structures within proof assistants.

References

[1] The Univalent Foundations Program, *Homotopy Type Theory: Univalent Foundations of Mathematics*, Institute for Advanced Study, 2013.

[2] Cyril Cohen, Thierry Coquand, Simon Huber, and Anders Mörtberg, *Cubical Type Theory: A Constructive Interpretation of the Univalent Foundations*, Journal of Automated Reasoning, 2018.

[3] Anders Mörtberg et al., *Agda Cubical Library*, `https://github.com/agda/cubical`, 2023.

[4] Michael Shulman, *Higher Inductive Types in Homotopy Type Theory*, `https://arxiv.org/abs/1705.07088`, 2017.